TravellingFL: Communication Efficient Peer-to-Peer Federated Learning

Vansh Gupta[†⋆], Alka Luqman[‡⋆], Nandish Chattopadhyay[‡], Anupam Chattopadhyay[‡], and Dusit Niyato[‡]

[†]Indian Institute of Technology, Delhi, India
[‡]School of Computer Science and Engineering, Nanyang Technological University, Singapore

***Abstract*—Peer-to-Peer federated learning is a distributed machine learning paradigm with a primary goal of learning a well-performing global model by collaboratively learning a shared model at different data hubs without the need of sharing data. Due to its immense practical applications, there is growing attention towards various challenges of efficient federated learning including communication efficiency, assumptions on connectivity, data heterogeneity, enhanced privacy, etc. In this paper, we address the problem of dynamic network topologies in federated learning. We present a technique to help new participants in Peer-to-Peer federated learning reach best possible accuracy by leveraging learning at other devices in a communication efficient manner. We model the costs in federated learning and apply a graph theoretical framework to show that one can draw from a range of graph-based algorithms to construct an efficient communication algorithm on a connected network, thereby matching the inference efficiency of centralized federated learning. We conduct experiments with varied graph formations and sizes to validate our claims.**

***Index Terms*—Communication Efficiency, Decentralized Federated Learning, Deep Learning (Machine Learning), Wireless communications, Internet of Things (IoT)**

I. Introduction

The modern world, in an attempt to simplify human tasks and enhance user experiences across a plethora of programs, sought to build systems, or rather, to inculcate the capability in systems of being able to imitate human cognizance. The field of computer science focused on strengthening this ability is what we know as machine learning. Computer scientists and statisticians have since pushed its boundaries - from using big data and bigger models trained on resource intensive servers, to smaller and scaled down models on edge devices. This led to the development of techniques that use aggregated data from various sources to effectively train a model and generate systems that can identify patterns or correlations within the data, which in turn helps make intelligent decisions with minimal human intervention. As the requirement for data is ever increasing, Federated Learning [1] proposes an alternate approach, which bypasses the centralized data requirement for incrementally learning from distributed data silos. We now have AI applications using federated learning on mobiles [2], smart home setups [3], unmanned aerial vehicles [4] etc. Deploying intelligence on edge devices, using peer-to-peer federated learning on resource constrained devices and networks, especially at scale, poses several challenges.

⋆*authors contributed equally*

A. Motivation

In a standard machine learning environment, there exists a centrally processed dataset in a tightly integrated system. Having all the data available simultaneously at one place is impractical as more often than not, data for large scale machine learning models is mined from different users, devices, and hubs - in silos, like medical data recorded at hospitals. Assembling all this rich data would require huge data centres and raise privacy concerns among the different data holders. Thus, to address this problem, [1] introduced an alternative that leaves the training data distributed on the mobile devices, and learns a shared model by aggregating locally-computed updates. They termed this decentralized approach 'Federated Learning'. Ideally, the models trained using federated learning are as good as a centralized model, or at the very least, better than what parties could learn individually.

There are two broad methods to implement Federated Learning. The first is a centralized method, where there is a central server which acts as a trusted coordinator among different data sources and the model is shared amongst the participating devices. The shared model is incrementally trained on each device's dataset and sent back to the server, which then sends the model parameters to the other nodes for further training. The second is a fully decentralized method which makes use of a peer-to-peer (P2P) framework. The devices are allowed to transfer the model directly with each other, within their capability, after training the model on their local dataset.

The bottleneck in both of these approaches becomes the computational power of each of the edge devices and their communication efficiency. Unless data is allowed to be shared, training will have to be done on the device itself, which makes the former compute problem less of a concern that we have significant control over with choice of devices. In this paper, we aim at solving or at least taking a step in the right direction, to resolve the latter issue of communication efficiency.

B. Contribution

In this paper, we present a new system-level perspective of fully decentralized federated learning. Our technical and conceptual contributions are as follows:

- **Systematization of Knowledge (SoK):** We put forward an SoK in the space of peer-to-peer federated learning

coupled with current SOTA standard problems and potential solutions, i.e., we provide a short critical analysis of previous approaches to the problems, list their limitations and then propose methods and future directions to mitigate these.
- **Graph Analogy:** We use a graph analogy as a working solution for a good representation of the system. This mapping allows one to utilize graph theory and its algorithmic tools to achieve the secondary goal of our work, i.e. efficient P2P communication.
- **Novel solution:** We capitalize on the advantages of federated learning and graphs to develop a solution to the problem at hand by modelling it as a known NP-hard problem and draw sub-optimal solutions from that domain which scale well. We look at the problem of providing optimal route for devices to communicate in a P2P setting as a modified version of the traveling salesman problem. The novelty is that we are able to frame the problem in a well-known standard form that can be solved efficiently by many available tools. We further implement a solution that jointly optimizes model accuracy with communication cost on the path travelled, achieving accuracy close to that of a model trained with traditional FL approaches [1] with significantly lesser communication overhead.

C. Organization of the Paper

Section II analyses and reviews the relevant works and literature connected to the various components that contributed to the building of the essence of our work. We also create a systematization of knowledge, condensing all the information in form of a textual tree, segregating related works to form unique clusters with their own benefits and drawbacks. Section III presents the system model and problem formulation. The details of the algorithm and infrastructure are present in Section IV, along with the methods used in the process. Section V details the implementation and experiments, including the experimental design, as well as the results obtained and the observations gained. In conclusion, Section VI touches upon the prospects of this work and what scope it creates for future contributions.

II. Background and Related Works

A. Federated Learning

Federated learning (FL) has attracted quite some attention from research ever since it was first introduced in 2016 regarding its extension, application and potential [1]. Recently, the main focus of enhancements have been in overcoming statistical problems [5], making it more personalized [5]–[7], and strengthening security & privacy [7]–[9]. The paper [10] summarises three challenges faced by FL as (1) device heterogeneity such as varying storage, computational and communication capacities; (2) statistical heterogeneity like the non-IID (a.k.a. non-independent and identically distributed) nature of data generated from different devices; (3) model heterogeneity, the situation where different devices want to customize their models adaptive to their application environments.

A recent article [11] gives the reader insights into the future research opportunities in FL suggesting some new possible directions revolving around data protection to look into. They also present a comparison between many different approaches and classify them into 6 broad categories namely-data distribution, machine learning model, privacy mechanism, communication architecture, scale of federation and motivation of federation.

The fine work in [12] highlights the need for personalization and surveys recent research in this area. As stated in [13], all the existing works achieve personalization in two separate steps that are associated with extra overhead. First, a global model is built, and then each client fine-tunes the global model using local data. While [12] mentions other techniques for building global models, including transfer learning, multi-task learning, meta-learning, and others; [13] proposes a novel algorithm by using LotteryFL where each client learns a lottery ticket network (i.e., a subnetwork of the base model) by applying the Lottery Ticket hypothesis, and only these lottery networks will be communicated between the server and clients. This way, instead of a global shared model as in classic federated learning, each client learns a personalized model all while reducing the communication cost.

The article [14] draws attention towards parameter updates for federated learning, which are crucial to ensure that models are working with the most recent parameters and questions the security of parameter gradient sharing. This work exploring adversaries in FL is essential for an effective implementation and acceptance of the popular technique.

B. Peer-to-Peer Federated Learning

A completely decentralized federated learning (DFL) has become an even more active area of research as the centralized approach potentially poses the same problems as traditional ML where all the data is at a single hub. Since our DFL setting makes use of the Peer-to-Peer (P2P) communication alone (previously talked about by [16]), we will only be covering works related to P2P DFL in this section. In a highly dynamic P2P environment, the goal is to have all centres directly talk with each other, instead of relying on the main server as per traditional federated learning. Such an application in the medical field is covered by [18].

The approaches in [24] and [25] aim at training personalized models, which is a different objective from ours as we try to find the single best shared global model. The former [24] provides an extensive analysis of the convergence rate, memory and communication complexity of their approach, and demonstrate its benefits compared to competing techniques on synthetic and real datasets. This approach is supplemented by [20] by making the algorithm differentially private to protect against the disclosure of information about the personal datasets, and formally analyze the trade-off between utility and privacy.

Gossip algorithms, as the name suggests, are a means of P2P communication in distributed systems by *leaking* infor-

TABLE I: Systemization of knowledge of P2P FL approaches on graphs

	Strongly connected network	**Network independent**
Synchronous	[15]Federated learning using peer-to-peer network for decentralized orchestration of model weights [16]Peer-to-peer federated learning on graphs [17]Decentralized federated learning for electronic health records [18]Braintorrent: A peer-to-peer environment for decentralized federated learning	[13]Lotteryfl: Personalized and communication-efficient federated learning with lottery ticket hypothesis on non-iid datasets [19]Deploy-able privacy preserving collaborative ml
Asynchronous	[20]Personalized and private peer-to-peer machine learning	[21]Personalized cross-silo federated learning on non-iid data [22]Edge-consensus learning: Deep learning on p2p networks with non-homogeneous data [23]Towards on-device federated learning: A direct acyclic graph-based blockchain approach

mation to the neighbours. These are a type of asynchronous algorithms and have been successfully employed in the area of decentralized optimization. Some variants of gossip algorithm have been put forward by [26], using dual averaging to solve problems both in synchronous and asynchronous settings and [27]. The study of adhoc teamwork [28] also tries to select agents to collaborate with from a distributed environment, in the absence of any prior knowledge.

C. Systemization of Knowledge

Broadly, any FL algorithm can be divided into different groups based on how centralized or decentralized a model's training approach is. We however focus on models that are entirely P2P, or attain personalized models in some way. This allows a degree of flexibility to works that are not fully decentralized but still offer a lot of insights to the field. First, the P2P FL approaches are segregated on the basis of network assumptions - works that assume a strongly connected network i.e., there exists a path in each direction between each pair of edge devices (vertices) of the network versus works that make no such presupposition. Among these groups, a further divide is made on the basis of synchronization - papers that explicitly state that their models are asynchronous and those which do not mention that or declare their model to be synchronous. This is categorized in Table I and elaborated below.

In the **first category**, which requires a strongly connected graph and works synchronously, there are many works, prominent of which are - [16], [18], [17] and [15]. The nodes in the proposed model by [16] take a Bayesian-like approach via the introduction of a belief over the model parameter space. The nodes update their belief by aggregating information from their one-hop neighbours to learn a model that best fits the observations over the entire network. Even though BrainTorrent [18] is particularly targeted towards medical applications, it presents a highly dynamic P2P environment that outperforms traditional server-based approaches and reaches a similar performance to a model trained on pooled data. In [15] a RAFT based aggregator selection is leveraged, introducing a dynamic aggregator which when coupled with dynamically generated cryptographic keys creates a more secure mechanism for delivery of models within the network while ensuring anonymity. In [17] the focus is on improving the communication efficiency for fully decentralized FL over a graph.

In the **second category**, with strongly connected network but asynchronous functioning, [20] in addition to solving the FL problem under mentioned constraints, provides a provable convergence rate. They also show how to make the algorithm differentially private to protect against the disclosure of information.

In the **third category**, the network constraints are loosened, which is more realistic but still requires synchronicity among nodes. A novel federated learning algorithm is presented in [29], where devices mostly collaborate with other devices in a pairwise manner, achieving 10X better communication efficiency. Strong guarantees of privacy with a marginal compromise in performance is shown in [19], which aims at preserving differential privacy of each participating client. They additionally experiment with quantization of model and discover that its deployment on edge devices does not degrade its capability.

Finally, the **fourth category** covers approaches that work with loosened network constraint and also works with asynchronously operating devices. Here, [21] facilitates learning between clients with similar data and [22] shows that effective deep learning is possible for data residing in different physical locations, even when these data are heterogeneous in nature, and without the need for centralized processing using an edge consensus learning method. A framework for empowering FL using Direct Acyclic Graph (DAG)-based blockchain systematically (DAG-FL) is introduced in [23].

D. Research Gaps

In this section, we will revisit some of the works discussed above and highlight opportunities that arise based on the state-of-the-art. It is worth noting that assuming a strongly connected network and having all the nodes to necessarily work synchronously are major research gaps in themselves. Practically, we cannot expect multiple edge devices (up to thousands in wireless sensor networks) to be directly connected to each other and manage to always be in a *ready* state.

In [20], each agent has a local clock ticking at the times of a rate 1 Poisson process, and it wakes up when its clock ticks. At this point, it updates the local model based on most recent information received from the neighbour. This somewhere limits the functioning because of the inability to accommodate multiple recent (and probably different) updates. To make

their algorithm differentially private for the edge devices, they replace the update step adding a noise vector drawn from a Laplace distribution with finite scale. If this value is constant throughout the training, as the model approaches the minima, the value of the updates will decrease, thereby increasing the percentage of noise, and thus deteriorating the accuracy-privacy trade-off.

In [15] a P2P solution to FL is presented as there is no single server, but this proposition only addresses one of the two problems of centralized FL. It does not rectify the issue of bandwidth as all the "messages" still go to a single node. Even though [17] lists data privacy as one of the challenges of FL, no data protection policy is incorporated in their model. Evidently, there is always a privacy-accuracy trade-off and same is the issue with [19]. While they state that precision is important and even use quantization to scale the weights, the added noise disrupts the precision.

Traditional efforts to find a communication efficient path in a wireless sensor network treat these goals in a decoupled manner, finding the best path with respect to either communication cost constraints (e.g., [30], [31], [32]) or ML task accuracy (e.g., [33], [34]). The work in [35] emphasises that FL navigation in vehicular fog must consider the communication overhead of ensuring privacy. They reduce communication costs by avoiding reconstructing keys of all participants and instead utilizing skip lists to limit the mutual key reconstruction to affected groups. We would like to look at making this a comprehensive decision involving communication and device capability constraints along with the task accuracy. We further consider how the presence of varying amounts of training data at each node conflicts with the choice of best path chosen according to only communication costs in a network.

While majority of FL works focus on improving the learning of the entire network, the primary question in this work is how a specific device can reach best possible accuracy with least possible communication cost. We consider this a pertinent question in networks with dynamic topologies.

E. Problem Scenarios

To discuss potential solutions to improve upon the communication aspect of P2P FL networks, it would be better to look at it from the application point of view and draw analogous parallels between mobility in wireless sensor networks and the task at hand. For this purpose, we consider the survey paper [36] that provides a comprehensive review on the developed methods that exploit mobility of sensor nodes and/or sink(s) to effectively maximize the lifetime of a mobile wireless sensor network (MWSN).

Vehicular Edge Computing (VEC) requires the training of models for object detection, lane detection, localization, mapping etc. [37] and using federated learning to train these models needs to operate in a dynamically changing environment. Cars are moving at different speeds in different directions which causes service switching among edge base stations or roadside units as well as switching the neighbourhood of the vehicle [38]. These vehicles can operate in areas with limited GPS connectivity where flexibility of FL participants is still problematic [35]. Autonomous vehicles need to train their local models by choosing the right vehicles to perform training on, which can lead to best accuracy in a resource-efficient and fast manner.

A similar problem exists while using Unmanned Aerial Vehicles (UAVs) over 5G wireless networks to perform disaster monitoring, goods delivery etc. in high mobility, high altitude conditions [39]. Network devices which lack sufficient training data to improve their accuracy above a certain threshold can effectively pass their model through other nodes in the network, utilizing the data residing at those nodes to accomplish their accuracy objectives. This is augmented with additional constraints, like limited channel bandwidth and power available at devices.

III. System Model and Problem Formulation

A. System Model

We have a set of N *data owners*, $\mathcal{N} = \{1, .., n, .., N\}$ that form the devices participating in the FL training rounds. Each device i has their own private *local dataset* $\mathcal{D}_i$. A device either initiates a model, if it is the starting node in the training round, or locally trains a received model for its own use as well as for sharing with its peer devices. In traditional machine learning, all local datasets are combined for model training $\mathcal{D}_N = \cup_{i \in \mathcal{N}} \mathcal{D}_i$, whereas in federated learning, a global model $\mathcal{M}_N$ is created by collecting and aggregating local models from all devices or from subset of devices $\mathcal{W}$ as $\cup_{i \in \mathcal{W}} \mathcal{M}_i$ where $\mathcal{W} \subseteq \mathcal{N}$. In our proposed scheme, TravellingFL, we do not enforce the presence of only one global model, instead, the model exists in different stages of training accuracy across the network. Some devices will be in possession of a nascent model, while others (present later in the best *path*) have a better trained or more accurate model. We assume, for now, that all nodes are honest and train on their local data and share their true local model. Note however that the proposed

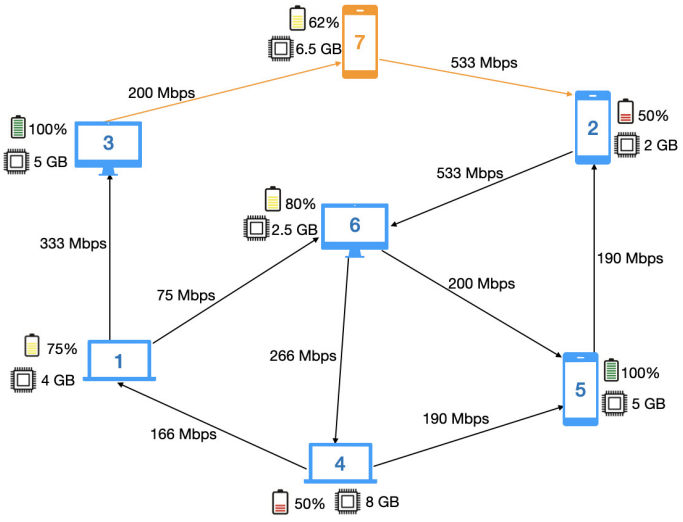

Fig. 1: System model of TravellingFL Simulation 1 with each device having its own local and private training data. Device 7 joins the existing network of 6 devices with different communication bandwidth channels to its peers.

TravellingFL can be extended for such a case by incorporating the node classification and anomaly detection methods.

We seek to address how to implement FL for dynamic topologies, specifically, how can a new device enter a network where FL may be ongoing or completed after reaching a stable accuracy, and bring itself up to best possible accuracy quickly while minimizing communication costs. In this way, multiple devices can train FL models along their individual optimal paths in parallel. This can easily be extended to multi-task FL [40] by having different TravellingFL paths in the same network apply to different ML objectives. It is important to note that the traditional approaches of FedAvg would perform averaging across all devices, including the new device which has limited learnt knowledge, thus bringing down the accuracy of the pre-existing devices, and then perform incremental learning all over again. This is not a desirable technique, as node dynamics should not affect the existing knowledge in the system.

While the primary task of FL is accuracy, the environment where edge devices are used to perform computations enforces a secondary goal, which becomes attaining that accuracy, or approaching it, with the lowest computations, communication costs, etc. This would allow to make the decentralized federated learning concept more realistically applicable. The system-level costs are divided into two categories - on-device compute costs and inter-device communication costs.

1) Compute Cost: The computation costs in FL are from the CPU cycles expended for each iteration of local model training, and are static in nature for a fixed number of training cycles on a given machine architecture. The data quality of each device is termed as the *node score*. It quantifies the power and memory available on a device, thereby reflecting if a device is ready to contribute to model training at each round or not. Device i's node score is denoted as

$$\mathcal{C}_i^N = f(D_i^p, D_i^m) = D_i^p * D_i^m \tag{1}$$

where D_i^p is the device i's remaining power level and D_i^m is its available free memory. The product of these two terms are taken, since devices with greater battery or power level and more amounts of available memory are preferred for model training. For a realistic model of device participation in FL, we consider a battery power of upto 50% as low, since personal devices can operate in power saving mode below that and choose to opt out of FL. We can also introduce a metric quantifying the quality of the local dataset at device i into this node score, such that devices with more training data or better quality data can be prioritized in the path selection.

2) Communication cost: The communication costs are more dynamic in nature, since they model the communication over a channel, wireless in this case, and depends on the power available at the edge device, bandwidth of the channel, etc. The *edge score* refers to this communication cost incurred by transferring the trained model across the channel and is denoted as

$$\mathcal{C}_{i,j}^E = f(C_{i,j}^p, C_{i,j}^b) = C_{i,j}^p / C_{i,j}^b \tag{2}$$

where $C_{i,j}^p$ is the transmission power required over the channel and $C_{i,j}^b$ is the channel bandwidth for edge e_{ij}. The greater the distance between nodes, more power is required during transmission and higher bandwidth channels will be prioritized since the graph formulation tries to minimize the sum of edge scores.

B. Problem Formulation

Given a network of interconnected devices with their own private data and varying communication constraints, our goal is to find how to achieve best possible inference accuracy on a target device within the shortest time. The constraints which apply to this problem are in terms of *compute* cost, which depends on the hardware specifications of the devices, and *communication* costs, which depends on the distance between devices (and thereby power expended for message transfer) and also inversely on the bandwidth of the communication channels. We impose a constant computation cost in our experiments, since all devices are created similarly. The communication cost is used as edge costs in the graph-based solution. To accurately model the working of an edge computing network, an efficient FL design must balance the primary goal of the network, which is the ML task, with the secondary goal, which is to intelligently utilize the underlying network's resources. Thus we jointly optimize the various costs along with the ML task accuracy.

The problem is formulated as a dual objective training - to minimize the categorical cross-entropy loss of the classification problem and the cost of communication along a selected path. The objective function is formulated as

$$\begin{aligned} \min_{K,e,n} \quad & \frac{1}{\mathcal{D}} \sum_{i=1}^{N} \sum_{d=1}^{\mathcal{D}_i} f(\mathbf{m}, \mathbf{x}_{id}, \mathbf{y}_{id}) * cost(K, \mathcal{C}^E, \mathcal{C}^N) \\ \text{s.t.} \quad & K \in \mathbb{Z}^+ \text{ and } \mathcal{C}^E, \mathcal{C}^N \in \mathbb{R}^+ \end{aligned} \tag{3}$$

where $\mathcal{D} = \sum_{i=1}^{N} \mathcal{D}_i$ is the total training dataset seen across all devices, $\mathbf{m}$ is the model trained on each local device, and $f(\mathbf{m}, \mathbf{x}_{id}, \mathbf{y}_{id})$ is the loss function used for attaining required model accuracy, which differs according to the ML task. Here we use categorical cross entropy as the loss function for the image classification task using data samples having input vector $\mathbf{x}_{id}$ and output vector $\mathbf{y}_{id}$ as

$$f(\mathbf{m}, \mathbf{x}_{id}, \mathbf{y}_{id}) = -\mathbf{y}_{id} log(\mathbf{x}_{id}^\top \mathbf{m}) + (1 - \mathbf{y}_{id}) log(1 - \mathbf{x}_{id}^\top \mathbf{m}) \tag{4}$$

The cost function $cost(K, \mathcal{C}^E, \mathcal{C}^N)$ is the expected cost incurred for model transfer along the path with K devices, where $\mathcal{C}^N$ is the device score of each of the K devices and $\mathcal{C}^E$ is the channel communication costs.

$$cost(K, \mathcal{C}^E, \mathcal{C}^N) = \alpha * \sum_{k=1}^{K-1} \mathcal{C}_{k,k+1}^E + \frac{\mathcal{C}_k^N}{2} + \frac{\mathcal{C}_{k+1}^N}{2} \tag{5}$$

The scores of nodes i and j which flank an edge e_{ij} are incorporated equally to reflect their united influence along a chosen path. Other works in the context of FL that define some concrete parameters for measurement of communication cost involve either transmitting or receiving on both devices. Some of these definitions can be found in [29], [23], [36] for further detail. We use a constant scalar α to normalize the cost

function and ensure equation (3) remains differentiable across all training rounds, which needs to be experimented with and appropriately set if the range of input costs changes.

The learning task generates a set of possible solutions called the Pareto optimal set, from which the best network is selected. The Pareto optimal set contains all the paths in the graph which can lead to model convergence for the defined ML task. We consider performing model training along all such contenders within a chosen heuristic distance to choose a best path, i.e. we explore the existence of a *slightly* longer path, within ϵ distance of the shortest path, which contains more or better training data, thus improving model training accuracy faster.

The training process takes the following approach. Step 1 is only performed once at the beginning of the exercise and Steps 3 and 4 are performed consecutively at each device.

1) *Task initialization* : Based on the number of devices participating in the FL round, the scheme maps the devices as nodes and assigns communication costs to the edges on the graph topology created. A set of possible best paths for traversal in the graph are chosen using Algorithm1 (detailed in Section IV) across all devices.
2) *Model initialization* : The starting node in the chosen path initializes a model $\mathbf{m}_i^0$ and the intermediate nodes instead of building a new model, will wait to receive a model from their peers to perform local learning.
3) *Local model training* : At iteration k, node i performs a round of training on model $\mathbf{m}_i^k$ using its local dataset $\mathcal{D}_i$ to generate local model $\mathbf{m}_i^{k+1}$ so as to minimize the predefined loss function $L(\mathbf{m}_i^k)$.
4) *Model transmission* : Once local model training is complete at iteration k, the model $\mathbf{m}_i^{k+1}$ is sent to the next node in the chosen traversal path.

The P2P FL *platform*, which we also refer to as the simulator in our experiments, is responsible for mapping the costs associated as edge weights in the graph formulation, and thus identifying the initial, intermediate and terminal nodes for each round of TravellingFL, after calculating the best path to be traversed. The parameters for costs calculated using equations (1) and (2) and the ranges from which they are uniformly drawn are listed in Table II. The details about the parameter distributions and allowed ranges are sourced from [41] and [42] for the 802.11n wireless standard. In this paper we assume that channel constraints remain constant across multiple training rounds, and hence the devices can perform multiple communication epochs under the same cost conditions.

TABLE II: Simulation Parameters

Parameter	Description	Values
N	Number of devices participating in FL	[2, 20]
$\|D_i\|$	Device i's Local dataset size	[1250, 5000] rows
$\|D_{test}\|$	Test dataset size	10000 rows
D_i^p	Device i's power level	[10, 100]%
D_i^m	Device i's available memory	[1, 10] GB
$C_{i,j}^p$	Channel transmission power over edge e_{ij}	[15, 24] dBm
$C_{i,j}^b$	Channel bandwidth over edge e_{ij}	[54, 600] Mbps
ϵ	Acceptable deviation from *shortest* path	10

IV. Graph-based Solution to Communication Efficient P2P FL

Now, we propose a novel algorithm to perform federated learning in a peer-to-peer framework which aims to accomplish two tasks- ML accuracy and communication efficiency as discussed in the following subsections.

Considering that each device brings its unique dataset to FL, it is desirable that all other devices learn from that data. In this setting, each device would have to perform model training on all other devices, and to perform this in a communication efficient manner, the problem translates to finding a least cost path across all nodes, i.e. Travelling Salesman Problem.

Our algorithm makes use of a graph analogy similar to [17] where the problem is mapped to a graph so as to enable us to make use of the entirety of graph theory for discovering solutions. As such, the participating devices are treated as nodes of a graph and the edges are communication links between them. When a new device enters an FL network where different devices possess their local NN models, 2 fundamental questions emerge -

1) Primary enquiry - Given there is one device in the network which has attained highest accuracy, along what path should that model be transferred to the new device such that communication costs are minimized? This is a trivial mapping to the open Traveling Salesman Problem with the constraint of visiting each device only once and which relaxes the constraint of creating a closed tour, thereby allowing us to find one-way routes.
2) Secondary enquiry – Is there an alternate path which may be slightly longer that attains a higher ML accuracy, as a virtue of it learning from more diverse training data at each device? We model ML accuracy as a minimization problem in Equation 4 and the tour cost as a minimization problem in Equation 5, this allows us to combine these 2 elements in Equation 3 to expand the previous TSP to an All Pair Shortest Path problem.

A. Algorithm Design

The network is a group of nodes (devices) connected by edges (communication links) possessing enough processing power to train a model on the data they carry. We plan to maintain a shared global model that is incrementally trained on each of the nodes and then the model is transferred to the next node, thereby *travelling* through the network. This can be applied for any connected graph, i.e., there is a path from any node to any other node in the network. The proposed algorithm solves for a directed graph, to allow for more generic cases where communication is not possible or entertained in both directions. This is useful to model devices which are broadcast-only nodes, or to capture the notion of trust among devices, which is not bidirectional, e.g. device A accepts a trained model from B, but B does not accept a model that comes from device A.

When a new node is added to a pre-existing FL network, the fastest way to improve the inference accuracy of the newly introduced node is to leverage the data available at all other devices. We can do this by applying transfer learning on a model which is exposed to the data at all other devices to get trained incrementally. This can be reformulated to the problem of visiting each node in the graph while minimizing the cost of travel, the integer linear programming formulation of which is

$$\min_{tour\ length} \quad \sum_{i=1}^{N} \sum_{j \neq i, j=1}^{N} C_{i,j}^{E} e_{ij} \tag{6}$$

where e_{ij} is a communication channel in graph G and $C_{i,j}^{E}$ is its corresponding communication cost as defined in 2. The comprehensive cost function defined in 5 is calculated using the shortest tour thus derived. In this way, the dual problem in equation 3 can be mapped to the Traveling Salesman Problem.

The graph is parametrizable, allowing for an easy way to calculate new paths through the network while tweaking device costs and connections. There must, however, exist at least one path, which when traversed on, will let the model be trained on the complete data present on all the nodes in the network. The edge weights are non-negative real numbers, and even if for some reason this is not the case, the values can be easily right-shifted or normalized. The goal of the problem is to balance between training an accurate model and keeping the communication efficient, therefore, the sum of edge scores (as defined before) along a route needs to be the least, while training a model on the datasets residing at the nodes. In a scenario where we want to train a new node by leveraging the learning present in an already fully trained network, the problem of finding the quickest path to bring the new node's model up to speed reduces to solving the shortest path across all nodes. In short, the communication efficiency part of the problem statement is *to find a least cost tour of a directed strongly connected graph visiting all nodes*.

To solve this problem, which would in turn present a solution for the task, we first considered the Travelling Salesman Problem (TSP), but it had some challenges associated, primary of which is that it is an NP-hard problem [43]. Secondly, there is no strict requirement to return to the origin of the path, and finally, there is no restriction to visit each "city" (node) just once as the model can *travel* through the network in a way that improves the model at the nodes visited. We can visit a node multiple times if there is no solution or no better solution the other way around. This involves performing at least one round of local training every time a node is visited, subject to device capability. The problem of this being NP-hard can be solved by adopting various heuristics like the ant algorithm, simulated annealing [44], closest neighbour, 2-opt [45], 3-opt [46], or Christofide's algorithm [47] to get a sub-optimal solution.

We consider the strategic shortest paths problem which while not exactly solving our problem, does offer an insight on a limitation that we will later address. The strategic shortest paths problem has a goal of sending a message between two points in a network but one has to keep in mind that the edges may have personalities where each of them have their own selfish interests. Thus, while discovering a path through the graph, a device may indicate that its transmission time is very long, in order to remain unbothered. A potential solution for this is to use the Vickrey-Clarke-Groves (VCG) mechanism [48] to incentivize device participation and avoid selfish routing.

Algorithm 1 Algorithm to find the Pareto optimal set of least cost paths for TravellingFL

Input: Graph G with node scores N and edge scores E
Output: Pareto optimal set of least-cost paths through the graph and cost of each path

$G \leftarrow$ Adjacency matrix with $G_{i,j} = C_{i,j}^{E}$
*/*Incorporate node costs into edge weight of graph*/*
Update $G_{i,j} = C_{i}^{N}/2 + C_{j}^{N}/2$ for valid edge e_{ij}
$S_{i,j} \leftarrow G_{i,j}$
*/*Finding All pairs Shortest path*/*
for $k = 1$ to n **do**
  for $i = 1$ to n **do**
    for $j = 1$ to n **do**
      if $(G_{i,j} > G_{i,k} + G_{k,j})$ **then**
        $S_{i,j} \leftarrow G_{i,k} + G_{k,j}$
      end if
    end for
  end for
end for
*/*To allow any node as starting point for best paths*/*
for $i = 1$ to n **do**
  for $j = 1$ to n **do**
    Add dummy $origin$ node connected to node i and node j with edge cost $C_{origin}^{E} = 0$
  end for
end for
*/*2-opt heuristic solution to find shortest paths*/*
for $i = 1$ to $n - 2$ **do**
  for $j = i + 2$ to n **do**
    $d1 \leftarrow C_{i,j}^{E} + C_{i+1,j+1}^{E}$
    $d2 \leftarrow C_{i,i+1}^{E} + C_{j,j+1}^{E}$
    if $(d1 > d2)$ **then**
      Swap j and $i + 1$
    end if
  end for
end for
$P \leftarrow$ All paths sorted in increasing order of total $cost$
$P_0 \leftarrow cost$ of first path in P
$K \leftarrow$ Subset of paths from P with $cost <= P_0 + \epsilon$
return K

We thus propose a new algorithm, which uses a 2-opt heuristic for the TSP, after tweaking the graph that mitigates the said limitations. Under this, the adjacency matrix of the network graph is created and the shortest distance between each pair of nodes is calculated and stored. This shortest paths matrix S is a strongly connected matrix as a result of an initial assumption that there necessarily exists one such path that involves every node. Before solving the TSP on this graph, a dummy origin node is set that is bi-directionally connected to all the nodes with edge weights 0. This node's outgoing edges

will make our work easier as it can be used to automatically decide which of the real nodes should be a starting point for the best path without significantly increasing the time complexity. The incoming edges enable the TSP solution to return to the origin without affecting the rest of the model. Once this graph is crafted, a 2-opt heuristic is used to solve TSP to get a tour which gives the order in which to visit the nodes in the original graph such that the total distance is minimized. The set of all such tours, P, are possible solutions for the least cost tours along which the model training can be performed, from which we select all candidate paths at a distance of ϵ, ϵ being a heuristic limit which is experimented with.

Using this method, the strongly connected graph assumption is loosened, there is no more restriction of returning back to the origin as the dummy is only a tool we use to get the path, and by using shortest distance matrix, the model that can visit each node multiple times if it produces a better solution. One important thing to note is that when a node is visited multiple times in the same round, it trains the model on its local data every time. This acts as an incentive for that device to expend its resources in transferring the model parameters to the next node, and if there are security layers, then decrypting and encrypting it for the next node. If a model gets trained more times on its data, it will be more personalized for that device.

An increase in number of participating devices introduces 2 facets to the FL objective.

1) In terms of connectivity – If the newly introduced devices introduce a greater number of communication channels, i.e. if they belong to the same sub-network or have some notion of trust between them, the underlying topology becomes a dense graph structure. In this context, increased interconnectivity leads to more paths between any 2 devices, consequently resulting in decreased number of model transfer hops. If the underlying topology is a sparse graph, the NN training can continue in parallel on the autonomous devices, but certain paths traversing nodes with higher in-degree or out-degree can experience congestion and consequently have a higher waiting time to receive the best model.
2) In terms of dataset diversity – An important aspect of FL is to train a model on non-i.i.d data which is a better representation of real-world data aggregation. With increased device participation, we can gain improved minority class representation and mitigate data bias, thereby allowing us to train a well-generalized ML model. In the TravellingFL formulation, we introduce a node score C_i^N in Equation 1 which can model the data quality at each device. If a global data quality assessment metric can be agreed upon in the network, this score can be used in the minimizing objective in Equation 5, for both i.i.d and non-i.i.d data, to further ensure that we choose paths that traverse devices with quality training data.

B. Algorithmic Analysis

The Floyd Warshall algorithm [49] is used to calculate the shortest path from any vertex to all other vertices thus implicitly storing $|V|^2$ shortest paths between the $|V|^2$ pairs of vertices in graph G, with added time of $O(|V|)$ to process the vertices on each path. The 2-opt heuristic tries to incrementally improve a path in the graph by making successive improvements that exchange two of the edges in the path with two other edges. This improvement step, which performs reversal of the sub-path produced by dropping edges can take $O(|V|)$ time. This is performed on all pairs of edges, which takes $O(|V|^2)$ time, thus making the overall complexity of this algorithm $O(|V|^3)$.

While $O(|V|^3)$ does look like a major design limitation, it is the maximum delay, and is still quite small in comparison to the time taken to train a fairly good ML model at the new node. Thus the time taken to calculate candidate paths is asymptotically negligible, especially for densely connected graphs. For large, sparse graphs we implement bounded search, whereby instead of exploring the entire graph space of $|V|$, the algorithm only looks at a subset of nodes that are within k steps of the starting vertex. Thus, the complexity reduces to $O(|k|^3)$ where k is the diameter of search, thus reducing the time taken in practice.

V. Experiments and Observations

This section details the experimental setup, design and results, followed by the key findings of the experiments conducted. We create a simulator for the Peer-to-Peer FL setup on which the experiments to validate our claims are performed. The motivation to build a simulator is to be able to quickly try out different configurations without the hassle and cost of acquiring physical devices to test on. The simulator helps to abstract out the complexities of setting up the network connectivity so that experimenting with different topologies and network channels only requires us to initiate new graphs and choosing a configuration setting. This also drastically reduces the initial time taken to create different experimental designs.

A. Experimental Setup

Docker containers[1] [50] are used to model independent devices which participate in model training, thus mimicking isolated edge devices in the distributed FL environment. Docker allows the use of different base images to closely align with the device which is to be replicated- it could be a simple Linux box, Android phone, or a Raspberry Pi device. The compute resources available to these devices can also be controlled using runtime options to limit memory, CPU and availability of GPU at the container level. Two methods of communication are implemented between the devices (Docker containers) - one using ZeroMQ[2] to transfer models among peer devices via ZeroMQ Transport Protocol (ZMTP) and another using NS3 ([51], [52]) to simulate model transfer via wireless networks. Currently the simulation supports modelling WiFi channel for transfer, but other channels like CSMA or 5G can be incorporated in the future, depending on the scenarios to model.

[1]https://www.docker.com/
[2]https://zeromq.org/

The capability to manipulate aspects of the communication channel like bandwidth is built using Linux's traffic control subsystem on each container and NS3's WiFi module. This enables us to further observe the relation between network parameters and model performance. All the Docker containers are managed using an orchestration service like Kubernetes. In order to seamlessly scale the number of devices in the setup, this simulator is deployed on a cloud computing platform like Amazon Web Services (AWS), with Amazon Elastic Kubernetes Service (EKS) used for orchestration.

The ability of the simulator to faithfully replicate real-world experiments is corroborated by running experiments from [53] on CIFAR-10 dataset where FedAvg algorithm [1] is used to combine individual device's trained models over 20 communication rounds to get similar reported accuracy of 85% on 7 clients and 87% on 10 clients. The TravellingFL simulation 1, described in detail later, is designed similarly on 7 devices and references the 85% accuracy as one of the baselines for performance. [53] reports the time taken for these experiments to be 31.3 minutes, and the simulator performs the same experiments in 28.7 minutes, showing that local model training which takes up the bulk of the experiment time remains unchanged, and only difference is in any hardware or setup overheads. In our experiments, we additionally control device-centric parameters like memory and GPU as well as network-centric parameters like channel bandwidth.

We evaluate our results on CIFAR-10 dataset which contains a total of 60000 color images belonging to 10 classes, 5000 training images and 1000 test images per class. This is partitioned randomly and uniformly across the devices participating in FL. In Simulation 1, each device contains 5000 training images per device and in Simulation 2, 1250 training images per device. We then conduct the TravellingFL experiments using a Simple MLP model built on base VGG-16 model with 4 additional activation layers followed by a final softmax classification layer. We build each device from Nvidia Tensorflow base image to use GPU capability during local model training. We experiment with uniform as well as varied bandwidths of communication channels between these devices. We experiment with varying number of local training epochs as well as communication rounds, and different local model architectures since we would like to evaluate model performance along with model size, as we look at the application of this setup in wireless sensor networks where resource limitations are a big constraint. Simulation 2 is a bigger network with 20 nodes initialized similarly, to observe the scalable nature of the algorithm. We are limited to 20 as the maximum number of devices in these experiments due to available quota restrictions on AWS at the time of experimenting.

B. Experimental Design

For our experiments, we define Simulation 1 as a graph network of 7 nodes and randomly initialised edge weights to indicate communication cost over a channel, as shown in Figure 1. The simulator spins up 7 Docker pods and establishes sockets for communication channels between peer devices, according to the edges defined. The communication cost is calculated as the sum of the weights of the edges traversed to send the model from one node to another.

The amount of training data present at each device is simulated to vary between 1250 and 5000 rows. We simulate the power and RAM availability of each device along with the communication channel's transmission power requirements and bandwidth. The range of values from which these parameters are sampled by the simulator are listed in Table II.

We use NS3 802.11 models to simulate the MAC and PHY implementations of WiFi using 1 access point and the individual devices as station nodes. The rate control algorithm used is the AARF algorithm [54], an adaptive algorithm for low latency systems. We also add mobility models such that station nodes are moving inside a 2-dimensional grid, using RandomWalk2dMobilityModel Helper in NS3, which moves the nodes in random directions at random speeds within the bounding box.

TABLE III: Comparison of test accuracy with prior work in Federated Learning. TravellingFL is able to achieve comparable accuracy even with reduced number of communications over the network.

FL Algorithm	FedAvg [1]	FedProx [55]	FedCAMS [56]	TravellingFL
Accuracy (%)	85	83.84	72.6	84.13

TABLE IV: Test set accuracy and communication cost trade-off of TravellingFL simulations over multiple number of communication rounds. Communication cost is calculated according to equation 2 for simulated network topology.

Experiment	**Number of rounds**	**Accuracy (%)**		**Comm Cost**
		1 Epoch	***2 Epochs***	*(x100)*
Baseline-1 [1]		79.4	85	43.2
Baseline-2 [55]		79.05	**83.84**	**35.4**
	1	70.72	78.3	4.35
TravellingFL	**5**	80.8	**84.13**	**27.15**
Simulation 1	10	85.15	84.78	55.6
	20	84.48	84.76	112.65
	40	84.98	84.86	226.65
Baseline [55]		80.85	**87**	**289.80**
	1	71.2	80.4	20.34
TravellingFL	2	77.85	82.09	43.56
Simulation 2	**5**	79.87	**85.67**	**113.22**
	10	83.05	86.05	229.32

C. Experimental Results

Table III conducts experiments of model training on simulation 1 (figure 1) over 5 communication rounds with 2 local training epochs on each device. The test set accuracy achieved is compared with prior works in the FL literature like the original FedAvg [1], FedProx [55] and FedCAMS using scaled

sign compressor [56], which is a communication efficient FL algorithm. Since we model the communication cost over a network regardless of size of the data transferred, we cannot compare the communication cost savings in a one-to-one manner with FedCAMS, which implements communication efficiency using compression techniques.

The communication cost incurred using FedProx is indicated as a baseline for further comparison in Table IV. It also summarizes different experiments run on TravellingFL simulations by varying number of local training epochs on each device, as well as training for multiple communication rounds. We can see that with just 2 local training epochs and 1 pass through the network, the model is able to converge to an accuracy of 78.3% with 87% reduction in communication cost. The model quality can be improved by performing multiple training passes through the network.

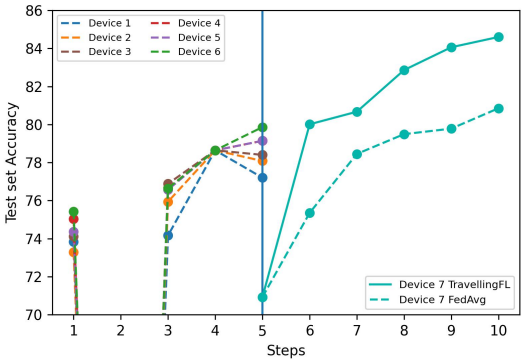

Fig. 2: Comparison of test accuracy of Device 7 using TravellingFL and FedAvg on Simulation 1 when pre-existing network has reached sufficient accuracy and is stable (denoted by vertical line at round 5) shows that TravellingFL avoids slow convergence by leveraging training at other nodes. In FedAvg, even indices indicate averaging whereas odd indices indicate local training epochs.

Effect of new device entering a stable network : TravellingFL can be used as the sole FL technique in a network or it can be used in conjunction with other FL techniques. We demonstrate the latter in Figure 2 by using FedAvg to train a group of devices until the network has reached sufficiently good accuracy with less variability amongst them, and then contrasting the number of steps taken to reach similar accuracy for this new device if we continue to use FedAvg or if we switch to TravellingFL. Device 7 is introduced in step 5, beyond which averaging occurs at steps 6, 8 and 10 to reach test accuracy of 80.8% via FedAvg, incurring communication cost of 5655 dBm/Mbps. In the case of TravellingFL, this takes 1 step of transferring a new model via the optimum path found to achieve test accuracy of 80.01% with communication cost of 2455 dBm/Mbps. Further training along the TravellingFL path shows faster convergence to best accuracy within the next 5 training epochs.

Effect of communication passes through network : Figure 3 shows the performance of TravellingFL on simulation1, both inference accuracy and loss respectively, with variation in number of communication passes through the network while keeping local training epochs fixed at 1. We see that the model

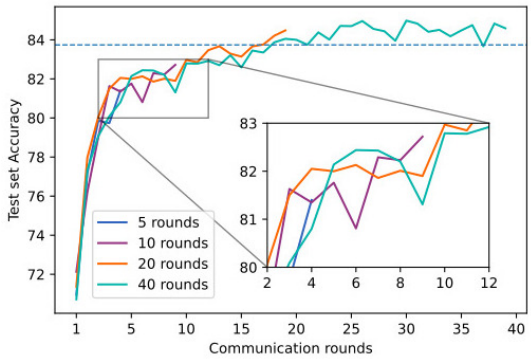

Fig. 3: Variation in accuracy with communication rounds by fixing local training epochs = 1. TravellingFL model is able to achieve high accuracy very quickly, even with smaller number of local training at each device. Increased passes through the network only improves the model marginally.

achieves a performance of 80.8% in just 5 communication passes, which improves to 84.48% with up to 20 communication passes. Further increasing the number of passes through the network results in diminished marginal returns as seen by a maximum accuracy of 84.98% reached over 40 communication passes through the network. This shows that TravellingFL is able to converge fast, thereby reducing costly communication bandwidth. The accuracy plateaus around 85% and this can be because each device only interacts with its immediate neighbour and the training data amongst these peers might not be diverse enough to include all classes [57].

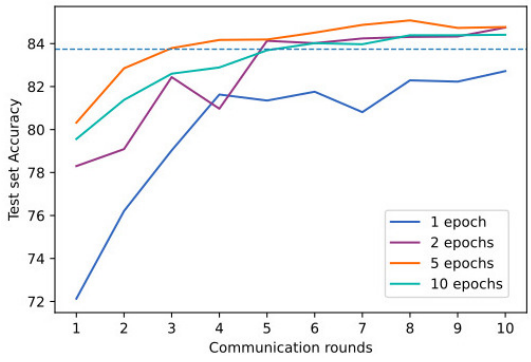

Fig. 4: Variation in accuracy with local training epochs. Even 2 rounds of local model training epochs can help with convergence.

Effect of local training epochs : Figure 4 shows the inference accuracy and loss respectively of TravellingFL with variation in number of local training epochs on simulation1 over 10 communication passes through the network. Performing more number of local training rounds is entirely dependent on the device level constraints and does not add to communication cost. More local training epochs do not necessarily translate to a better model as seen in Figure 6 that 10 local epochs performs at par with 5. But it allows the devices to start from a higher accuracy, as seen in Figure 5. This can be taken advantage of by devices participating later in the training to start converging from a higher accuracy, thus allowing for lesser number of communication passes overall,

as seen in Figure 5.

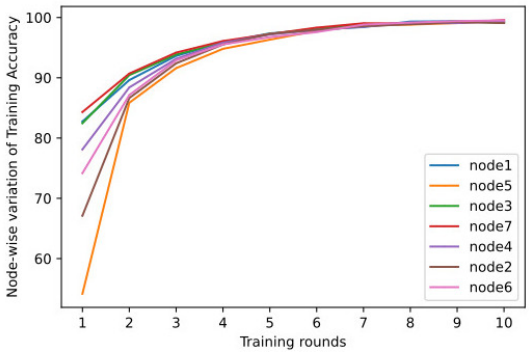

Fig. 5: Variation of training accuracy at each node. Devices appearing later in the travelling path are able to leverage training accuracy achieved at prior nodes to reach model convergence.

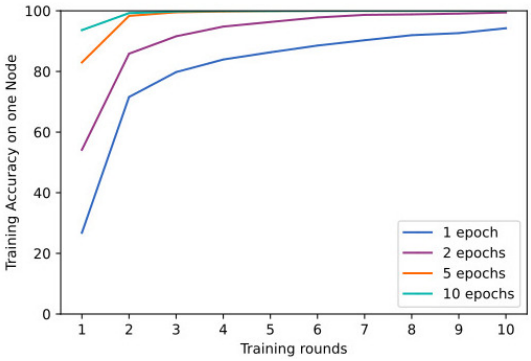

Fig. 6: Variation of training accuracy at one node with different number of local epochs. There is not much to gain by increasing number of local training epochs at each device beyond an optimum.

We observed in our experiments that the normalization value used in the cost functions needs to be carefully chosen so that the gradients of the joint optimization function remain derivable. This means that if the input range of the simulation parameters are changed, the exercise to identify the normalization value must be performed again.

Shortest path vs Optimal path : Figure 7 plots the different paths under consideration while solving the joint optimization problem. The *shortest path* is path0 and other paths are part of the Pareto optimal set, within chosen ϵ distance of 10. We can see that path2 is able to achieve better accuracy than path0, after 5 communication rounds of 5 local training epochs each, highlighting the importance of *jointly* considering ML task accuracy and network communication constraints. We run this experiment setting over 5 communication passes through the network and report the performance in Table V. Reduced number of local training exaggerates this difference but shows a similar convergence pattern in Figure 8, with the shortest path defined by communication constraints alone leading to a sub-optimal model.

Scalability : Figure 9 plots the accuracy variation for both simulation 1 and simulation 2 across 2 local training epochs. We can see that the algorithm behaves similarly even on scaling the number of devices in the topology.

TABLE V: Accuracy achieved by shortest path vs best path chosen by TravellingFL with varying data distribution at nodes

Path	**Accuracy (%)**		**Comm cost**
	1 Epoch	***5 Epochs***	*(x100)*
Shortest path (path0)	71.68	80.07	27.15
TravellingFL path (path2)	79.26	80.13	27.90

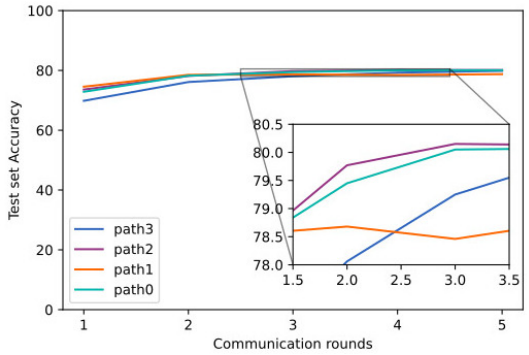

Fig. 7: Variation of test accuracy achieved along different paths in the Pareto optimal set, with 5 local training epochs. Shortest path through the network is path0 and TravellingFL chosen optimum path is path2.

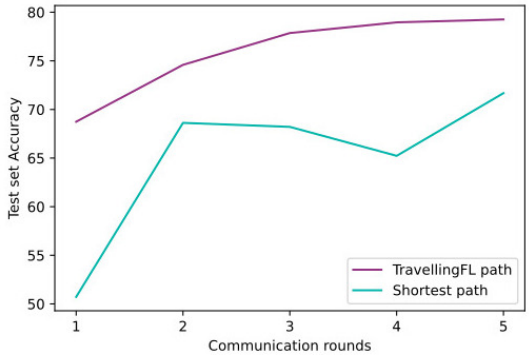

Fig. 8: Comparison of test accuracy achieved along shortest path vs TravellingFL path with reduced compute of 1 local model training epoch shows difference in accuracies achieved.

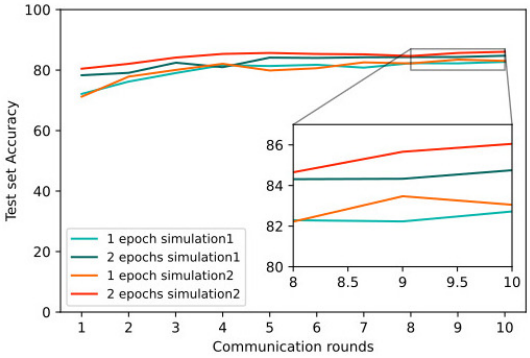

Fig. 9: Comparison with larger simulation network to observe scalable nature of TravellingFL shows similar model convergence behaviour.

Performance of different DNN architectures : Figure 10 plots accuracy and loss respectively of simulation 1 with different model architectures, which is also summarized in

TABLE VI: Various model architectures implemented in the simulation

Model architecture	VGG 16	VGG 19	Custom VGG	ResNet 50	SqueezeNet
Model size (MB)	128	170.6	138.3	215.6	6.7
Accuracy (%)	85.28	84.62	84.48	78.86	68.79

Table VI. We experiment with different CNN architectures like VGG-16, VGG-19 [58] and ResNet-50 [59] to see the impact of number of trainable parameters on model performance. The ResNet-50 model has more than double the number of layers but we can see that it achieves accuracy of 78.6% as opposed to 85.28% using VGG-16 model. The complexity of these models means it takes more time to train its parameters, and also bigger size of the trained model which is transferred over the channel. So we look at SqueezeNet [60] which has a smaller network architecture and was shown to achieve AlexNet levels of accuracy despite being a much smaller model. This is highly favourable in resource constrained domains like wireless sensor networks. But the lesser number of parameters also means we were able to achieve 68.48% accuracy at best using SqueezeNet version1 with simple bypass.

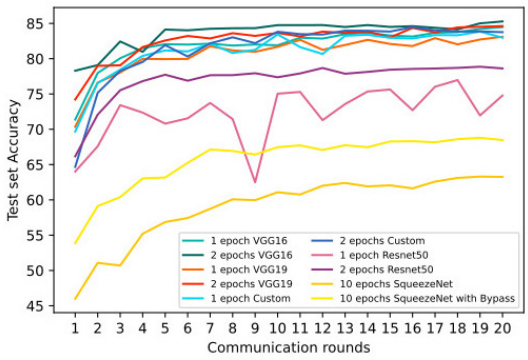

Fig. 10: Comparison of training accuracy with different model architectures. Achievable accuracy for models with limited number of parameters, like SqueezeNet, is bounded. Signs of model moving away from global minima are readily rectified within one full network pass, as indicated by the recovery from training accuracy drop in 1 epoch Resnet50.

Effect of training data size at each device : To explore the impact of size of training data on TravellingFL convergence, we perform multiple training experiments, each with different but uniform dataset sizes at each node and plot them in Figure 11 to show variation in accuracy. In the case of lesser data to learn from, more number of local training epochs prove beneficial for the model to converge faster, but there remains a gap in final inference accuracy on the test set, which is not bridged even after 20 communication rounds. This means that if there is not much variation in the data present across devices, then there is only so much the model can learn even after multiple training rounds across all the devices.

We experiment with available bandwidth of the communication channel between the devices - an unconstrained 100 Mbps channel taking 2 seconds for model transfer, a constrained 1 Mbps channel for all communication links taking 16 minutes on average for model transfer, and a varied topology with available bandwidth sampled uniformly from 54 to 600 Mbps, which is the data rate range for 802.11n wireless standard [42]. While Figure 12 shows the time taken for transferring model over these channels, Figure 13 also shows that this has no impact on the model performance as we are not modelling any packet losses during the transfer.

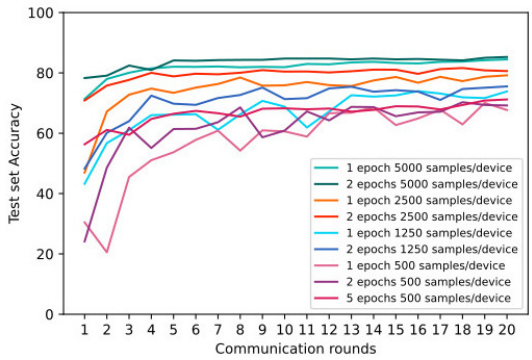

Fig. 11: Comparison of accuracy with varied training dataset size at each node. Less training data impedes model convergence and thus number of local training epochs and network passes can be chosen accordingly.

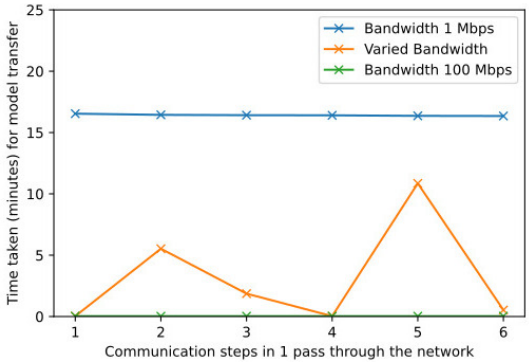

Fig. 12: Time taken for model transfer over bandwidth-constrained communication channels. Time taken for transfer is in accordance with communication costs modelled.

D. Key Findings

- We demonstrate that we are able to achieve inference accuracy similar to other distributed FL techniques with much lower communication cost using TravellingFL. The choice of best path for model training in a network must be holistically chosen based on data, device and channel constraints which apply to the defined ML task.
- Increasing the number of local training epochs is beneficial when there is lesser training data available at each device. Increase in number of local epochs beyond a point increases training accuracy at each device, thereby overfitting on locally available data, or being heavily personalized to that node. But this does not necessarily translate to biased global model performance, as infer-

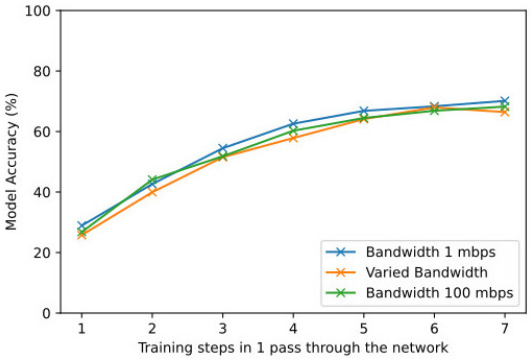

Fig. 13: Model convergence on bandwidth-constrained experiments, since we do not model lossy transfer.

ence accuracy does not increase beyond an achievable maximum.

- Devices appearing later along the training path get a more well-trained model, sometimes even at first participation in the training exercise.
- The communication cost, when measured in terms of time, could be reduced by 23.3% on average for the various device models, and communication structures that we experimented with. It remains an interesting theoretical study to find the lower and upper bounds of achievable cost savings with TravellingFL.

VI. Conclusion and Future Directions

A comprehensive survey of categorizing federated learning algorithms based on synchronization of models and network connectivity assumptions is presented in this work. Firstly, an overview of federated learning, Peer-to-Peer federated learning and the importance of optimizing communication costs in large scale peer networks is provided. Then, the basic tools for formulating the dual problem of performing model training under multiple constraints of communication network and computation power are derived. Afterwards, a graph analogy and corresponding implementation for dynamic networks, reviews and analyses for TravellingFL is given. We clearly show the importance of system-level perspective in approaching Peer-to-Peer federated learning challenges through a wide range of experimental studies.

We present, to the best of our knowledge, the first formulation to model system behaviour in Peer-to-Peer federated learning. We also outline some future directions for the consideration of the research community. Beyond these studies, one can model the overhead costs associated with model compression techniques to further reduce communication costs, and for implementing privacy in the proposed model to observe its impact on model convergence. The security of the communication network can also be investigated with respect to attacks like Denial of Service (DoS). While we modelled device costs at a compute level, aspects related to the quality of local data present at these devices can also be included to see if it improves the choice of best path.

Acknowledgment

This research is supported by the National Research Foundation, Singapore under its Strategic Capability Research Centres Funding Initiative, Infocomm Media Development Authority under its Future Communications Research & Development Programme, DSO National Laboratories under the AI Singapore Programme (AISG Award No: AISG2-RP-2020-019), Energy Research Test-Bed and Industry Partnership Funding Initiative, Energy Grid (EG) 2.0 programme, DesCartes and the Campus for Research Excellence and Technological Enterprise (CREATE) programme, and MOE Tier 1 (RG87/22). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Singh and J. Zhu, Eds., vol. 54. PMLR, 20–22 Apr 2017, pp. 1273–1282. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html

[2] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," 2018. [Online]. Available: https://arxiv.org/abs/1811.03604

[3] C. Zhou, A. Fu, S. Yu, W. Yang, H. Wang, and Y. Zhang, "Privacy-preserving federated learning in fog computing," *IEEE Internet of Things Journal*, vol. 7, no. 11, pp. 10 782–10 793, 2020.

[4] F. Qi, X. Zhu, G. Mang, M. Kadoch, and W. Li, "Uav network and iot in the sky for future smart cities," *IEEE Network*, vol. 33, no. 2, pp. 96–101, 2019.

[5] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/6211080fa89981f66b1a0c9d55c61d0f-Paper.pdf

[6] F. Chen, Z. Dong, Z. Li, and X. He, "Federated meta-learning for recommendation," *CoRR*, vol. abs/1802.07876, 2018. [Online]. Available: http://arxiv.org/abs/1802.07876

[7] M. Ammad-ud-din, E. Ivannikova, S. A. Khan, W. Oyomno, Q. Fu, K. E. Tan, and A. Flanagan, "Federated collaborative filtering for privacy-preserving personalized recommendation system," *CoRR*, vol. abs/1901.09888, 2019. [Online]. Available: http://arxiv.org/abs/1901.09888

[8] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 1175–1191. [Online]. Available: https://doi.org/10.1145/3133956.3133982

[9] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *CoRR*, vol. abs/1712.07557, 2017. [Online]. Available: http://arxiv.org/abs/1712.07557

[10] Q. Wu, K. He, and X. Chen, "Personalized federated learning for intelligent iot applications: A cloud-edge based framework," *IEEE Open Journal of the Computer Society*, vol. 1, pp. 35–44, 2020.

[11] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, Y. Li, X. Liu, and B. He, "A survey on federated learning systems: Vision, hype and reality for data privacy and protection," *IEEE Transactions on Knowledge and Data Engineering*, p. 1–1, 2021. [Online]. Available: http://dx.doi.org/10.1109/TKDE.2021.3124599

[12] V. Kulkarni, M. Kulkarni, and A. Pant, "Survey of personalization techniques for federated learning," in *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, 2020, pp. 794–797.

[13] A. Li, J. Sun, B. Wang, L. Duan, S. Li, Y. Chen, and H. Li, "Lotteryfl: Personalized and communication-efficient federated learning with lottery ticket hypothesis on non-iid datasets," *CoRR*, vol. abs/2008.03371, 2020. [Online]. Available: https://arxiv.org/abs/2008.03371

[14] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients - how easy is it to break privacy in federated learning?" in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 16 937–16 947. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/c4ede56bbd98819ae6112b20ac6bf145-Paper.pdf

[15] M. R. Behera, s. upadhyay, S. Shetty, and R. Otter, "Federated learning using peer-to-peer network for decentralized orchestration of model weights," Mar 2021. [Online]. Available: https://www.techrxiv.org/articles/preprint/Federated_Learning_using_Peer-to-peer_Network_for_Decentralized_Orchestration_of_Model_Weights/14267468/1

[16] A. Lalitha, O. C. Kilinc, T. Javidi, and F. Koushanfar, "Peer-to-peer federated learning on graphs," *CoRR*, vol. abs/1901.11173, 2019. [Online]. Available: http://arxiv.org/abs/1901.11173

[17] S. Lu, Y. Zhang, and Y. Wang, "Decentralized federated learning for electronic health records," in *2020 54th Annual Conference on Information Sciences and Systems (CISS)*, 2020, pp. 1–5.

[18] A. G. Roy, S. Siddiqui, S. Pölsterl, N. Navab, and C. Wachinger, "Braintorrent: A peer-to-peer environment for decentralized federated learning," *CoRR*, vol. abs/1905.06731, 2019. [Online]. Available: http://arxiv.org/abs/1905.06731

[19] N. Chattopadhyay, R. Maiti, and A. Chattopadhyay, "Deploy-able privacy preserving collaborative ml," in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, 2020, pp. 1397–1402.

[20] A. Bellet, R. Guerraoui, M. Taziki, and M. Tommasi, "Personalized and private peer-to-peer machine learning," in *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Storkey and F. Perez-Cruz, Eds., vol. 84. PMLR, 09–11 Apr 2018, pp. 473–481. [Online]. Available: https://proceedings.mlr.press/v84/bellet18a.html

[21] Y. Huang, L. Chu, Z. Zhou, L. Wang, J. Liu, J. Pei, and Y. Zhang, "Personalized cross-silo federated learning on non-iid data," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 9, pp. 7865–7873, May 2021. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/16960

[22] K. Niwa, N. Harada, G. Zhang, and W. B. Kleijn, "Edge-consensus learning: Deep learning on p2p networks with nonhomogeneous data," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 668–678. [Online]. Available: https://doi.org/10.1145/3394486.3403109

[23] M. Cao, L. Zhang, and B. Cao, "Toward on-device federated learning: A direct acyclic graph-based blockchain approach," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2021.

[24] V. Zantedeschi, A. Bellet, and M. Tommasi, "Fully decentralized joint learning of personalized models and collaboration graphs," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, S. Chiappa and R. Calandra, Eds., vol. 108. PMLR, 26–28 Aug 2020, pp. 864–874. [Online]. Available: https://proceedings.mlr.press/v108/zantedeschi20a.html

[25] P. Vanhaesebrouck, A. Bellet, and M. Tommasi, "Decentralized Collaborative Learning of Personalized Models over Networks," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Singh and J. Zhu, Eds., vol. 54. PMLR, 20–22 Apr 2017, pp. 509–517. [Online]. Available: https://proceedings.mlr.press/v54/vanhaesebrouck17a.html

[26] I. Colin, A. Bellet, J. Salmon, and S. Clémençon, "Gossip dual averaging for decentralized optimization of pairwise functions," in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1388–1396. [Online]. Available: https://proceedings.mlr.press/v48/colin16.html

[27] A. Koloskova, S. Stich, and M. Jaggi, "Decentralized stochastic optimization and gossip algorithms with compressed communication," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 3478–3487. [Online]. Available: https://proceedings.mlr.press/v97/koloskova19a.html

[28] R. Mirsky, I. Carlucho, A. Rahman, E. Fosong, W. Macke, M. Sridharan, P. Stone, and S. V. Albrecht, "A survey of ad hoc teamwork research," 2022.

[29] L. Chou, Z. Liu, Z. Wang, and A. Shrivastava, "Efficient and less centralized federated learning," in *Machine Learning and Knowledge Discovery in Databases. Research Track*, N. Oliver, F. Pérez-Cruz, S. Kramer, J. Read, and J. A. Lozano, Eds. Cham: Springer International Publishing, 2021, pp. 772–787.

[30] B. Luo, X. Li, S. Wang, J. Huang, and L. Tassiulas, "Cost-effective federated learning in mobile edge networks," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 12, pp. 3606–3621, 2021.

[31] J.-H. Chang and L. Tassiulas, "Maximum lifetime routing in wireless sensor networks," *IEEE/ACM Transactions on networking*, vol. 12, no. 4, pp. 609–619, 2004.

[32] Z. Zhou, S. Yang, L. Pu, and S. Yu, "Cefl: Online admission control, data scheduling, and accuracy tuning for cost-efficient federated learning across edge nodes," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9341–9356, 2020.

[33] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, B. McMahan *et al.*, "Towards federated learning at scale: System design," *Proceedings of Machine Learning and Systems*, vol. 1, pp. 374–388, 2019.

[34] A. Imteaj and M. H. Amini, "Distributed sensing using smart end-user devices: Pathway to federated learning for autonomous iot," in *2019 International conference on computational science and computational intelligence (CSCI)*. IEEE, 2019, pp. 1156–1161.

[35] Q. Kong, F. Yin, R. Lu, B. Li, X. Wang, S. Cui, and P. Zhang, "Privacy-preserving aggregation for federated learning-based navigation in vehicular fog," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 12, pp. 8453–8463, 2021.

[36] L. Nguyen and H. T. Nguyen, "Mobility based network lifetime in wireless sensor networks: A review," *Computer Networks*, vol. 174, p. 107236, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128619315865

[37] L. Liu, S. Lu, R. Zhong, B. Wu, Y. Yao, Q. Zhang, and W. Shi, "Computing systems for autonomous driving: State of the art and challenges," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6469–6486, 2020.

[38] H. Xiao, J. Zhao, Q. Pei, J. Feng, L. Liu, and W. Shi, "Vehicle selection and resource optimization for federated learning in vehicular edge computing," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[39] A. Fotouhi, H. Qiang, M. Ding, M. Hassan, L. G. Giordano, A. Garcia-Rodriguez, and J. Yuan, "Survey on uav cellular communications: Practical aspects, standardization advancements, regulation, and security challenges," *IEEE Communications surveys & tutorials*, vol. 21, no. 4, pp. 3417–3442, 2019.

[40] T. Zeng, J. Guo, K. J. Kim, K. Parsons, P. Orlik, S. Di Cairano, and W. Saad, "Multi-task federated learning for traffic prediction and its application to route planning," in *2021 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2021, pp. 451–457.

[41] J. S. Ng, W. Y. B. Lim, Z. Xiong, X. Cao, J. Jin, D. Niyato, C. Leung, and C. Miao, "Reputation-aware hedonic coalition formation for efficient serverless hierarchical federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 11, pp. 2675–2686, 2022.

[42] S. Banerji, "On IEEE 802.11: Wireless LAN technology," *CoRR*, vol. abs/1307.2661, 2013. [Online]. Available: http://arxiv.org/abs/1307.2661

[43] C. H. Papadimitriou, "The euclidean travelling salesman problem is np-complete," *Theoretical Computer Science*, vol. 4, no. 3, pp. 237–244, 1977. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0304397577900123

[44] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.220.4598.671

[45] S. Lin, "Computer solutions of the traveling salesman problem," *Bell System Technical Journal*, vol. 44, no. 10, pp. 2245–2269, 1965.

[46] A. S. Alfa, S. S. Heragu, and M. Chen, "A 3-opt based simulated annealing algorithm for vehicle routing problems," *Computers & Industrial Engineering*, vol. 21, no. 1, pp. 635–639, 1991. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0360835291901653

[47] H.-C. An, R. Kleinberg, and D. B. Shmoys, "Improving christofides' algorithm for the s-t path tsp," *J. ACM*, vol. 62, no. 5, nov 2015. [Online]. Available: https://doi.org/10.1145/2818310

[48] L. Makowski and J. M. Ostroy, "Vickrey-clarke-groves mechanisms and perfect competition," *Journal of Economic Theory*, vol. 42, no. 2, pp. 244–261, 1987. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0022053187900871

[49] Risald, A. E. Mirino, and Suyoto, "Best routes selection using dijkstra and floyd-warshall algorithm," in *2017 11th International Conference on Information & Communication Technology and System (ICTS)*, 2017, pp. 155–158.

[50] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux journal*, vol. 2014, no. 239, p. 2, 2014.

[51] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena, "Network simulations with the ns-3 simulator," *SIGCOMM demonstration*, vol. 14, no. 14, p. 527, 2008.

[52] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," in *Modeling and tools for network simulation*. Springer, 2010, pp. 15–34.

[53] A. Mathur, D. J. Beutel, P. P. B. de Gusmão, J. Fernández-Marqués, T. Topal, X. Qiu, T. Parcollet, Y. Gao, and N. D. Lane, "On-device federated learning with flower," *CoRR*, vol. abs/2104.03042, 2021. [Online]. Available: https://arxiv.org/abs/2104.03042

[54] M. Lacage, M. H. Manshaei, and T. Turletti, "Ieee 802.11 rate adaptation: a practical approach," in *Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, 2004, pp. 126–134.

[55] A. K. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith, "On the convergence of federated optimization in heterogeneous networks," *arXiv preprint arXiv:1812.06127*, vol. 3, p. 3, 2018.

[56] Y. Wang, L. Lin, and J. Chen, "Communication-efficient adaptive federated learning," *arXiv preprint arXiv:2205.02719*, 2022.

[57] T. Sun, D. Li, and B. Wang, "Decentralized federated averaging," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–12, 2022.

[58] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014. [Online]. Available: https://arxiv.org/abs/1409.1556

[59] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

[60] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size," *CoRR*, vol. abs/1602.07360, 2016. [Online]. Available: http://arxiv.org/abs/1602.07360