Graph-Based Traffic Forecasting via Communication-Efficient Federated Learning

Chenhan Zhang*‡, Shiyao Zhang*, Shui Yu‡, James J.Q. Yu*,
*Dept. of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China.
†Academy for Advanced Interdisciplinary Studies, Southern University of Science and Technology, Shenzhen, China.
‡School of Computer Science, University of Technology Sydney, Sydney, Australia.
zhangch@mail.sustech.edu.cn, syzhang@ieee.org, shui.yu@uts.edu.au, yujq3@sustech.edu.cn

2022 IEEE Wireless Communications and Networking Conference (WCNC) | 978-1-6654-4266-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/WCNC51071.2022.9771883

***Abstract*—The existing Federated Learning (FL) systems encounter an enormous communication overhead when employing GNN-based models for traffic forecasting tasks since these models commonly incorporate enormous number of parameters to be transmitted in the FL systems. In this paper, we propose a FL framework, namely, Clustering-based hierarchical and Two-step-optimized FL (CTFL), to overcome this practical problem. CTFL employs a divide-and-conquer strategy, clustering clients based on the closeness of their local model parameters. Furthermore, we incorporate the particle swarm optimization algorithm in CTFL, which employs a two-step strategy for optimizing local models. This technique enables the central server to upload only one representative local model update from each cluster, thus reducing the communication overhead associated with model update transmission in the FL. Comprehensive case studies on two real-world datasets and two state-of-the-art GNN-based models demonstrate the proposed framework's outstanding training efficiency and prediction accuracy, and the hyperparameter sensitivity of CTFL is also investigated.**

***Index Terms*—Federated learning, communication efficiency, graph neural networks, traffic forecasting.**

I. Introduction

In the era of big data, people's travel experience significantly benefits from real-time and accurate traffic states which is derived by massive recorded data [1]. To achieve higher accuracy, data-driven approaches such as machine learning (ML) and deep learning (DL) have been preferred in traffic forecasting tasks. Among various data-driven approaches, Graph Neural Networks (GNN)-based approaches are acknowledged as the *state-of-the-art*, which are well-suited to the spatial extracting of transportation networks and contribute to the successes of GNN-based methods in traffic forecasting.

Commonly, the training of conventional GNN-based models is centralized, which creates significant privacy concerns when multiple data providers are involved. Federated Learning (FL) has arisen as a solution to the privacy concerns associated with centralized training. FL protects data providers' privacy by enabling distributed training without requiring original data transfer and by storing data providers' data locally.

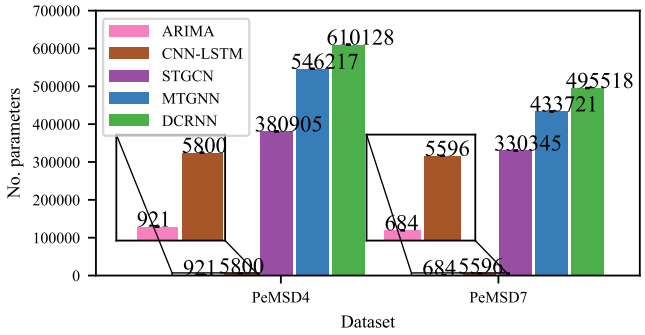

Fig. 1. Number of incorporated parameters of traffic prediction models on different datasets. STGCN [2], MTGNN [3], and DCRNN [4] are GNN-based models.

While the majority of optimization methods for FL systems are developed and evaluated using traditional ML/DL learning models [5], it is not surprising to generalize the FL to GNN-based models. Nonetheless, much research studies on GNNs and FL demonstrate a practical concern. Specifically, for FL, the majority of FL systems adopt the averaging algorithm (i.e., FedAvg [6]) to develop the global model, which requires multiple participants to upload their local models to the central server. This results in a significant increase in network communication costs and storage requirements for the central server. Compared with the conventional DNN models that handle regular grid-like data, the GNNs that handle high-dimensional and sparse graph-structured data is more computationally intractable and, at the same time, require more parameters as shown in Figure 1. The employment of the GNN-based models will significantly increase the overall computing and communication costs to FL systems [7]. Additionally, for existing data collection nodes in intelligent transportation systems, such as roadside units (RSUs), and mobile devices, may operate in places with unreliable networks. Such network environments pose a challenge for achieving real-time data processing in any FL systems. Existing research towards the communication optimization of FL includes gradient compression [8], asynchronous update [9], etc. However, as indicated above, most of these approaches are developed and tested considering conventional ML/DL tasks and models, the gen-

This work is supported by the Stable Support Plan Program of Shenzhen Natural Science Fund No. 20200925155105002, by the Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation No. 2020B121201001, and by Australia ARC DP200101374 and LP190100676. James J.Q. Yu is the corresponding author.

eralizability of which is dubious. Furthermore, there are few feasible communication-efficient solutions that merge GNN-based traffic forecasting approaches into FL. These approaches either result in a considerable amount of information being lost during the compression process or are incompatible with large-scale and real-time systems.

To fill this research gap, we propose a novel FL framework, named CTFL, for GNN-based spatial-temporal traffic forecasting tasks in this paper. In CTFL, we adhere to the concept of dispersed FL and propose a hierarchical FL architecture and a clustering approach for grouping clients. Specifically, the proposed clustering algorithm is based on the local model similarity of the clients, and we seek to group clients with comparable model parameters into the same cluster. Then, we integrate particle swarm optimization (PSO) algorithm that devises a two-step approach for local model optimization. Specifically, this technique seeks to determine the optimal local model through fitness evaluation, which requires the client to upload only a measured fitness value to the server installed in each cluster rather than the model parameters. As a result, the CTFL allows for uploading only one representative local model update from each cluster to the central server, significantly reducing the communication cost associated with model update transmission in the FL system.

The highlights of this paper are summarized below:

- We propose a communication-efficient framework, CTFL, for traffic speed forecasting tasks under FL.
- A parameters similarity-based clients clustering is proposed based on parameters similarity computation.
- A two-step local model optimization approach is designed based on the PSO algorithm.
- Comprehensive case studies on two real-world traffic datasets are conducted to demonstrate the efficacy of the proposed CTFL framework.

The rest of this paper is organized as follows. We elaborate on the proposed CTFL framework in Section II. Section III presents the results and discussion of the case studies. This paper is concluded in Section IV with a summary of potential future studies.

II. Methodology

A. Problem Definition

1) Traffic Forecasting on Graphs: We represent a transportation network as an undirected graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$, where $\mathcal{V}$ is the set of nodes and each of them is defined as a road segment, $\mathcal{E}$ is the set of edges, and $\mathcal{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ is the adjacency matrix of $\mathcal{G}$. $\forall\ v_i, v_j \in \mathcal{V}$, we have $[\mathcal{A}_{i,j}] = 1$ if v_i and v_j are connected, and 0 otherwise, where $[\mathcal{A}_{i,j}]$ is the entry of $\mathcal{A}$ that represents the connectivity between node v_i and node v_j. This is a common formulation among spatial-temporal traffic forecasting [2], [4]. The traffic data observed on $\mathcal{G}$ is denoted as a graph-wide feature matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}|\times\mathcal{F}}$ where $\mathcal{F}$ is the dimension of involved features (i.e., traffic records) of each node. Let vector $\mathbf{X}^t \in \mathbb{R}^{|\mathcal{V}|}$ denote the traffic data observed at time $t \in \mathcal{F}$, the objective is to learn a predictor $f(\cdot)$ that can develop graph-wide traffic predictions $\hat{\mathbf{X}}^{t+1}, \hat{\mathbf{X}}^{t+2}, \ldots, \hat{\mathbf{X}}^{t+s}$ in the following s time stamps (i.e., the prediction time horizon is s), given historical traffic observations of T stamps (i.e., the historical time horizon is T).

2) Federated Learning Scenario: In this paper, a practical FL scenario is considered for the proposed framework. Specifically, we define $\mathcal{G}$ as the converted graph of the entire transportation network of an area. This area is divided and conquered by multiple organizations[1] (e.g., companies, governments, and individuals). Let $\mathcal{C} = \{C_1, C_2, \ldots, C_M\}$ denote the organization set where M is the number of organizations. Each organization maintains a number of sensor stations (i.e., multiple nodes in the graph) that collect data on traffic.

To enable each local organization to *fully* exploit the spatial correlations, all of them can access the complete topological information of $\mathcal{G}$ freely but keep their respective traffic data unrevealed to others for privacy concerns. Furthermore, this study is based on the common assumption that there are no overlapping sensor stations between any two organizations [10], [11]. The central server distributes a copy of the adopted GNN-based model to each organization. The organizations treat the received model as local models and train them *simultaneously* and locally utilizing the stored training data and the topology. Saying "*simultaneously*" is because we only focus on the *synchronous update* that the organizations communicate with the server regularly for model updates by synchronous updates.

B. Framework Overview

As illustrated in the lower half of Fig. 2, CTFL involves two phases, namely, the organization clustering phase and FL phase. In the organization clustering phase, following a "divide-and-conquer" strategy, the organizations are clustered into multiple clusters by their learned spatial-temporal information. This clustering algorithm will be presented in Section II-C. In the FL phase, we propose a novel Particle Swarm Optimization (PSO)-based algorithm for local model training, which will be detailed in Section II-D.

C. Parameters Similarity-based Organizations Clustering

Because of the datasets' homogeneity in time-series and spatial correlations, the difference between the learned parameters of their models is indistinctive for some organizations. Aggregating homogenous models cannot contribute to developing a generic global model, which also incurs a superfluous communication overhead by transmitting such homogenous model parameters to the central server, especially when there is a large number of participated organizations [12]. To address this issue, inspired by the similarity-based clustering algorithms proposed in [13], and the hierarchical and dispersed FL structure introduced in [14], we propose a parameter similarity-based approach to clustering the organizations.

Specifically, the clustering decision is determined by the similarity of organizations' model parameters. Considering

[1]In this paper, "organization" and "client" are used interchangeably.

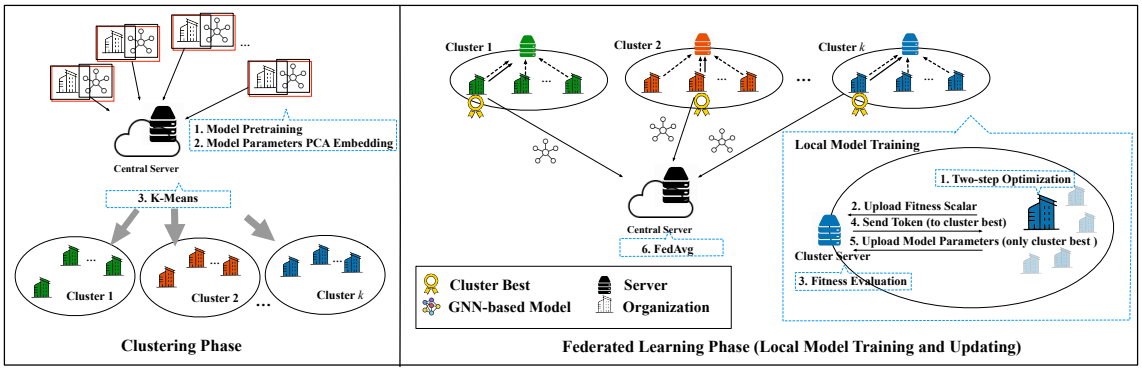

Fig. 2. The schematic of CTFL. In the Federated Learning Phase, the solid line represents the model parameters transmission while the dash line represents the fitness scalar transmission.

the high dimensional characteristics of the GNN-based model parameters, we adopt Principal Component Analysis (PCA) to reduce the dimension of model parameters and the K-means algorithm to achieve the clustering. To obtain the model parameters for similarity computing, we first pre-train the local models of each organization. For organization C_i and its data $\mathbf{X}_i \in \mathbb{R}^{|\mathcal{V}|\times\mathcal{F}}$, we use a portion of its historical data $\mathbf{X}_i^* \in \mathbb{R}^{|\mathcal{V}|\times\mathcal{F}^*}$ which is randomly sampled from the time series windows, satisfying $\mathcal{F}^* \ll \mathcal{F}$ is the sampled dataset[2]. Subsequently, $\mathbf{X}_i^*$ is used to train the local model, and we regard the learned model parameters θ_i^* as the pre-learned model parameters. Treating the model parameters as a high-dimensional vector $\theta_i^* \to \vec{\theta}_i^h$, we use PCA to project the high-dimensional $\vec{\theta}_i^h$ into a lower-dimensional space:

$$\vec{R}_i = \mathrm{PCA}_{\sum\sigma^2}(\vec{\theta}_i^h), \tag{1}$$

where $\vec{R}_i$ is the lower-dimensional vector of θ_i^* after dimension reduction; $\mathrm{PCA}(\cdot)$ denotes the PCA process and $\sum\sigma^2$ is the summative variance of all individual principal components[3].

Next, following the K-means algorithm, given a number of clusters k satisfying $k \ll M$, we randomly select k organizations as the initial cluster centroids. For each organization, we seriatim compute the similarity between its model parameters vector and the cluster centroids' ones. Particularly, for any two organizations C_i and C_j, the distance between their lower-dimensional model parameter vectors $\vec{R}_i$ and $\vec{R}_j$ is defined as

$$\mathrm{sim}_{i,j} := \mathrm{sim}(\vec{R}_i, \vec{R}_j) := \sqrt{\sum_{l=1}^{d}(\vec{R}_i[l] - \vec{R}_j[l])^2}, \tag{2}$$

where d is the dimension of $\vec{R}$ and l is the index number. Whereafter, we assign the organization to the cluster whose centroid has the highest parameter similarity with it. This optimization proceeds iteratively until the optimal solution is found, i.e., the assignments no longer change.

[2]The number of random sampling time points, i.e., F^*, are set the same for each organization.

[3]We empirically set $\sum\sigma^2 = 0.9$ to avoid information loss.

D. Two-step Local Model Parameters Optimization

We devise a two-step parameter optimization approach integrating particle swarm optimization (PSO) algorithms [15] for model training in CTFL. PSO is an iterative optimization algorithm, which has comprehensive capacities for global optimization along with easy execution, fast convergence, and robustness. PSO has a lower computational cost due to the use of simple mathematical operators, both in terms of speed and memory [16]. The PSO algorithm incorporates a group of particles, namely, a swarm. Each particle is considered a solution to the global problem, with a specified position, speed, and fitness value determined by an optimization function. Specifically, the particle's speed controls the next location it will reach. The fitness value is used to determine the particle's performance. To get the global optimal solution, the particles interact with one another frequently in order to fine-tune themselves.

Step 1: Therefore, in a local training epoch, we first use PSO to optimize the model parameters iteratively:

$$\nabla\mathrm{PSO}: \begin{cases} \vec{v}_{i,j}^{\tau+1} = \omega\cdot\vec{v}_{i,j}^{\tau} + \vec{U}(0,\phi_1)\cdot(\vec{\mathrm{pb}}_{i,j} - \vec{v}_{i,j}^{\tau}) \\ \qquad\qquad + \vec{U}(0,\phi_2)\cdot(\vec{\mathrm{cb}}_j - \vec{v}_{i,j}^{\tau}) \\ \vec{\theta}_{i,j}^{\tau+1} = \vec{\theta}_{i,j}^{\tau} + \vec{v}_{i,j}^{\tau+1} \\ \vec{\mathrm{pb}}_{i,j} = \vec{\theta}_{i,j}^{\tau+1} \end{cases}, \tag{3}$$

where i,j denotes the i-th client in the j-th cluster; τ denotes the τ-th iteration; ω denotes the inertia constant; $\vec{v}_{i,j}^{\tau}$ denotes the position of the particle of $C_{i,j}$ at iteration τ and thus $\vec{v}_{i,j}^{\tau+1}$ can be regarded as the computed speed at iteration τ; $\vec{\mathrm{cb}}_j$ represents the "cluster best" of cluster P_j; $\vec{U}(0,\phi_1)$ and $\vec{U}(0,\phi_2)$ denotes two vectors of random values uniformly sampled from $[0,\phi_1]$ and $[0,\phi_2]$ respectively, which are initialized at each iteration for each particle. We denote the pso-optimized model parameter by $\vec{\theta}_{i,j}^{\mathrm{pso}}$.

Step 2: Subsequently, we further optimize $\vec{\theta}_{i,j}^{\mathrm{pso}}$ by a conventional gradient descent approach iteratively, which can be formulated as

$$\nabla\mathrm{GD}: \vec{\theta}_{i,j}^{\mathrm{pso}} \leftarrow \vec{\theta}_{i,j}^{\mathrm{pso}} - \eta\cdot\nabla\mathcal{L}(\vec{\theta}_{i,j}^{\mathrm{pso}}, \mathcal{A}, \mathbf{X}_{i,j}). \tag{4}$$

We finally use $\vec{\theta}_{i,j}^{\mathrm{ts}}$ to denote the two-step-optimized model parameter for client $C_{i,j}$'s local model .

TABLE I
PREDICTION ACCURACY COMPARISON.

MAPE(%)/	STGCN		MTGNN	
RMSE	PeMSD4	PeMSD7	PeMSD4	PeMSD7
Centralized	4.75/4.78	6.93/5.11	4.72/4.93	7.02/5.33
FedAvg	4.79/4.82	6.90/5.13	4.73/4.90	6.99/5.29
FedQSGD	8.31/7.16	15.56/8.30	5.73/5.55	11.62/7.19
FedTopK	16.70/11.44	29.02/13.76	11.51/8.92	23.27/11.45
CTFL	4.84/4.87	7.08/5.25	4.79/4.91	7.08/5.23

To find the optimal local model, we introduce a fitness scalar among the clients, which is obtained by the forecasting performance of the two-step optimized local model on a section of randomly sampled data, and the performance is measured by Mean Absolute Percentage Error (MAPE) (See Eq. (7)). After receiving all the fitness scalars from clients, the cluster server then performs a fitness evaluation that seeks for the smallest one among the scalars, and the corresponding client denotes the fittest organization. Subsequently, the cluster server asks the fittest client to upload its model parameters (by sending a token) to the central server and store it temporarily as $\vec{\mathrm{cb}}^*{}_j$.

According to the divide-and-conquer strategy of CTFL, a sub-global model with $\vec{\theta}_j^{\text{best}}$ is first developed for different clusters by fitness evaluation. Thus, we can directly select the local model developing the smallest fitness scalar as the sub-global model rather than aggregating all local models. From the perspective of communication optimization, only one client is asked to upload the model parameters in such a case, which obviously reduces the overall communication cost. Then, we aggregate the sub-global models from different clusters in the central server to update the global model using FedAvg:

$$\vec{\theta}^{\text{global}} = \frac{1}{k}\sum_{j\in k}\vec{\theta}_j^{\text{best}}, \tag{5}$$

where $\vec{\theta}^{\text{global}}$ denotes the updated model parameters of the global model. Thereafter, in the model broadcasting step, instead of directly assign the global model to the clients, we let the cluster server to average the previously stored $\vec{\mathrm{cb}}^*{}_j$ and the received global model by

$$\vec{\mathrm{cb}}_j = \frac{1}{2}(\vec{\mathrm{cb}}^*{}_j + \vec{\theta}^{\text{global}}). \tag{6}$$

Then, the cluster server broadcasts $\vec{\mathrm{cb}}_j$ to included clients for the next iteration of local model optimization. Thus, we send an aggregation of the global update and the cluster best to the clients, making the PSO algorithm work more smoothly [17].

III. EXPERIMENTS

A. Dataset and Configurations

In the case studies, two real-world datasets are adopted for investigation, i.e., PeMSD4 and PeMSD7. PeMSD4 and PeMSD7 are two public datasets published by California Department of Transportation[4], which contains traffic speed data of Bay area from 307 sensor stations in the first two months of 2018 and Los Angeles from 228 sensor stations in the May and June of 2012, respectively. We develop the traffic graph topology of the two datasets based on sensor stations' distance following [2]. The data points in both two datasets are with a 5-min sampling interval. We adopt Z-score to normalize the speed values and linear interpolation to recover missing data points. For each dataset, the training, validation, and test sets are correspondingly constructed, each containing 60%, 20%, and 20% of all data, respectively. For traffic speed forecasting, the past time window is set to 60 minutes, and we use them to predict the speed in the next 45 minutes. MAPE and RMSE are used as the accuracy metrics. In particular, MAPE is considered as a preferable one (see [18], [19] for examples), which is defined as

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{X_i - \hat{X}_i}{X_i}\right| \times 100\%, \tag{7}$$

where X_i and $\hat{X}_i$ are observed and predicted traffic speeds at time i, respectively.

We include two state-of-the-art GNN-based models in our case studies: **1) STGCN** [2] which is a GNN-based and CNN-based model which integrates graph convolutions with 1D convolutions; **2) MTGNN** [3] which is a GNN-based and CNN-based model which leverages mix-hop propagation, adaptive graph, and dilated inception to exploit spatial-temporal dependencies. The hyperparameters and architectures of the two models, as well as the optimizers used, are based on the best settings accessible in the relevant literature.

Unless other stated, for fairness, we set the global epoch $E_{\text{global}} = 30$ and the mini-batch size $B = 50$ for all case studies. Furthermore, we set the number of local training epochs $E_{\text{local}} = 1$ for all the FL-based approaches to make them compared with the centralized training approach fairly. To simulate the FL training scenario for CTFL, we construct the respective dataset of different organizations. In particular, we first partition the whole traffic graph into M sub-graphs for M organizations using Metis partitioner [20]. Then we select the traffic data of included sensors (nodes) in a sub-

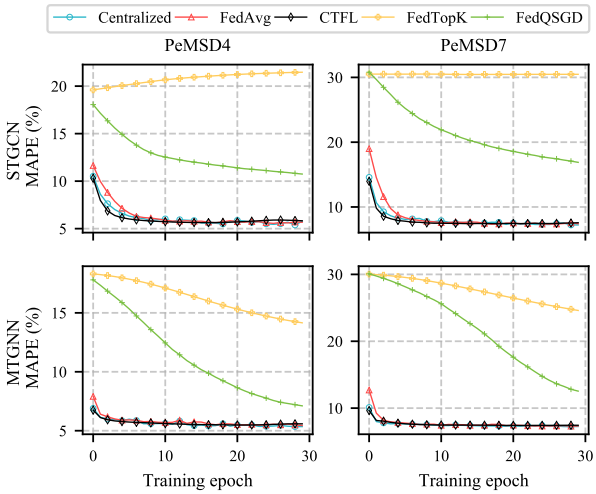

Fig. 3. Learning curves of CTFL and baselines.

[4]https://pems.dot.ca.gov/

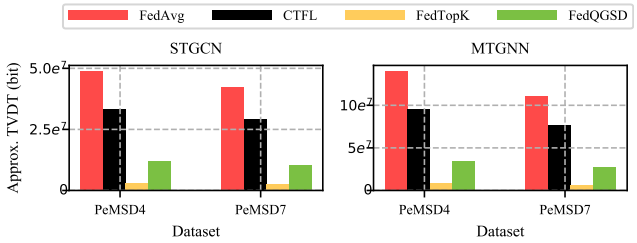

Fig. 4. Total volume of data transmission (TVDT) in a global epoch.

graph and construct the dataset for the corresponding organization. Due to the constraints of computing resources, we let $M = 8$, $k = 3$ (for CTFL). The default setting of PSO is $\omega, \phi_1, \phi_2 = 0.1, 1, 4$. CTFL and the baseline approaches are implemented with PyTorch using half-precision (i.e., float16) tensors in this work.

B. Learning Performance Comparison

In this case study, we configure the aforementioned two GNN-based models to CTFL and evaluate the traffic forecasting accuracy performance on the two datasets, respectively. We also compare the performance between CTFL with four baseline approaches: **1) Centralized**: Centralized training algorithm (i.e., training without FL); **2) FedAvg**: Naive unweighted FedAvg algorithm [6]; **3) FedQSGD**: FedAvg algorithm integrating with gradient compression [8], and QGSD 4-bit is adopted in our simulations; **4) FedTopK**: FedAvg algorithm integrating with k-sparsification for gradient sparsification [21], and $k = 1$ for sparsification is adopted in our simulations.

Table I summarizes the prediction accuracy results of CTFL and baselines. Figure 3 presents the corresponding learning curves, and Figure 4 illustrates the data transmission volume for the FL-based approaches. A few conclusions can be drawn from the results. First, the GNN-based models can obtain satisfying prediction accuracy with the proposed CTFL framework, which is comparable to FedAvg and centralized approaches. This is attributed to the learning efficacy of the proposed two-step local model updating approach in CTFL; the integrated PSO algorithm can efficiently calibrate the local models with global optimal, while the conventional GD algorithm can further guide the local models in the correct direction of convergence. Furthermore, as shown in Figure 4, achieving such accuracy performance, CTFL develops lower communication overhead compared to FedAvg; this proves that CTFL can achieve a significant trade-off between communication overhead and accuracy performance.

While FedQSGD and FedTopK approaches can incur even fewer communication overhead (See Figure 4) due to their adopted gradient compression and sparsification algorithms, we can find obviously poor convergence and accuracy performance developed by these two approaches. Especially for the FedTopK approach, the model fails to converge when adopting STGCN on PeMSD7. This is due to that remarkably spatial-temporal information learned and embedded by the model parameters are undermined during the processes of gradient

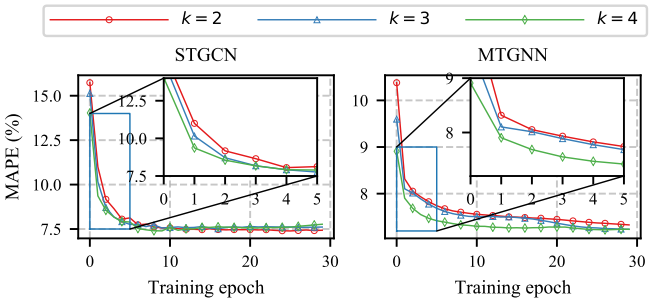

Fig. 5. The sensitivity of learning performance to cluster number k.

compression/sparsification when using these approaches. Although we find promising results presented in their original literature [8], [21], it is worth noticing that these results are developed using simpler or Euclidean modeling-based approaches (e.g., logistic regression and CNN-based models). That is to say, unless there is further well-pointed optimization, these gradient compression- or sparsification-based approaches cannot achieve promising results using GNN-based models on at least complex tasks such as traffic forecasting.

C. Ablation Test

*1) **Hyperparameter Sensitivity Test**:* We first assess the learning performance against the number of clusters k[5]. In this simulation, we set $k = 2, 3, 4$ for CTFL and compare the learning performance under this group of settings. From the simulation results shown in Figure 5, we find that the smaller number of involved clusters, the slower the learning curves converge. This result can be explained by the fact that a larger number of clusters can accelerate the convergence speed since the global model can aggregate learned information from more local models according to the proposed scheme.

In addition to the number of clusters k, it is also interesting to investigate the impact of the hyperparameters of the PSO algorithm in CTFL, i.e., inertia constant ω, acceleration constants ϕ_1 and ϕ_2. Specifically, we let $\omega \in \{0.05, 0.1, 0.2\}$, $\phi_1 \in \{0.5, 1, 2, 3, 4\}$ and $\phi_2 \in \{2, 3, 4, 5, 6\}$ to conduct a grid-search to identify the best portfolio. The simulation results[6] imply that CTFL is limitedly sensitive to the different selection of hyperparameters in the PSO algorithm. In particular, the MAPE ranges for the two models are $6.90\% - 7.20\%$ (STGCN) and $6.84\% - 7.14\%$ (MTGNN). While the grid-search results do not demonstrate a clear pattern to determine a better hyperparameters portfolio, such fine-tuning work can slightly improve the framework performance.

*2) **Variants Test**:* In the proposed framework, two core approaches, namely, parameters similarity-based organizations clustering and two-step local model parameters optimization, are orchestrated to achieve communication-efficient and accurate collaborative traffic forecasting. In this case study, we conduct a test to identify the contribution of these components to the overall framework performance. Particularly, two

[5]Note that since the performance results on two datasets demonstrate a similar pattern, for conciseness without loss of generality, we only present the framework performance on PeMSD7 dataset in the following simulations.

[6]Due to the page limitation, the complete result of this test is not presented.

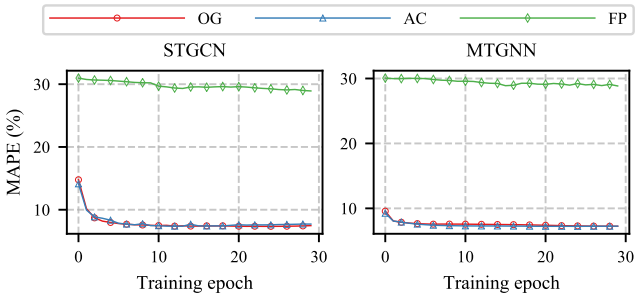

Fig. 6. Learning performance comparison between CTFL and its variants.

variants are constructed based on the original CTFL design as follows:

- CTFL-AC (Adaptive-Clustering): The proposed clustering approach is implemented iteratively in the FL phase by the central server.
- CTFL-FP (Full-PSO): In the proposed two-step optimization for local models, the GD optimization is removed, and only the PSO is used (i.e., one-step optimization).

The original CTFL framework is denoted by CTFL-OG in this case study. Figure 6 shows the simulation results.

Comparing CTFL-OG and CTFL-AC, we observe that the performance difference is minuscule; nonetheless, the latter's design renders it costing more communication and computing overhead due to its repetitive clustering computation. This result indicates that the design of one-off clustering in the proposed approach can provide the same level of performance as the iterative one with less overhead. Comparing CTFL-OG and CTFL-FP, we can observe that the variant CTFL-FP does not converge. The PSO algorithm cannot solely drive the model to correct convergence direction without the support of gradient descent, which implies the necessity of a two-step optimization strategy in CTFL.

IV. Conclusion

In this paper, we propose a novel FL framework for collaborative GNN-based spatial-temporal traffic forecasting. Compared with the existing FL frameworks, the proposed CTFL employs a dispersed and hierarchical architecture by using a clustering approach to cluster the clients into different groups based on their parameters' similarity. We also devise a two-step approach for the local models' optimization in the proposed framework, which follows the idea of particle swarm optimization. This scheme enables the model parameters from only one client of each cluster to be uploaded to the central server, which remarkably cuts down on the communication cost of model update transmissions in the FL. The case studies indicate that, compared with the baselines, CTFL can achieve the trade-off between efficient model communication and accurate model performance. In the future, we will involve more GNN-based models and datasets in the case studies to evaluate the generalizability of the proposed framework.

References

[1] P. Sun and A. Boukerche, "Security enhancing method in vehicular networks by exploiting the accurate traffic flow prediction," in *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, IEEE, 2021.

[2] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 3634–3640, 2018.

[3] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "Connecting the dots: Multivariate time series forecasting with graph neural networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 753–763, 2020.

[4] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *International Conference on Learning Representations*, 2018.

[5] Y. Liu, S. Zhang, C. Zhang, and J. James, "Fedgru: Privacy-preserving traffic flow prediction via federated learning," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–6, IEEE, 2020.

[6] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*, pp. 1273–1282, 2017.

[7] L. Ma, Z. Yang, Y. Miao, J. Xue, M. Wu, L. Zhou, and Y. Dai, "Towards efficient large-scale graph neural network computing," *arXiv preprint arXiv:1810.08403*, 2018.

[8] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "Qsgd: Communication-efficient sgd via gradient quantization and encoding," *Advances in Neural Information Processing Systems*, vol. 30, pp. 1709–1720, 2017.

[9] M. R. Sprague, A. Jalalirad, M. Scavuzzo, C. Capota, M. Neun, L. Do, and M. Kopp, "Asynchronous federated learning for geospatial applications," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 21–28, Springer, 2018.

[10] J. Feng, C. Rong, F. Sun, D. Guo, and Y. Li, "Pmf: A privacy-preserving human mobility prediction framework via federated learning," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, no. 1, pp. 1–21, 2020.

[11] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *IEEE Transactions on Wireless Communications*, 2020.

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

[13] F. Sattler, K.-R. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[14] M. S. H. Abad, E. Ozfatura, D. Gunduz, and O. Ercetin, "Hierarchical federated learning across heterogeneous cellular networks," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8866–8870, IEEE, 2020.

[15] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, vol. 4, pp. 1942–1948, IEEE, 1995.

[16] B. Qolomany, M. Maabreh, A. Al-Fuqaha, A. Gupta, and D. Benhaddou, "Parameters optimization of deep learning models using particle swarm optimization," in *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 1285–1290, IEEE, 2017.

[17] S. Bansal, E. Ghaderi, C. Puglisi, and S. Gupta, "Neural-network based self-initializing algorithm for multi-parameter optimization of high-speed adcs," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 1, pp. 106–110, 2020.

[18] C. Zhang, S. Zhang, J. James, and S. Yu, "Fastgnn: A topological information protected federated learning approach for traffic speed forecasting," *IEEE Transactions on Industrial Informatics*, 2021.

[19] H. Yao, F. Wu, J. Ke, X. Tang, Y. Jia, S. Lu, P. Gong, J. Ye, and Z. Li, "Deep multi-view spatial-temporal network for taxi demand prediction," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[20] G. Karypis and V. Kumar, "Metis: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices," 1997.

[21] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified sgd with memory," *arXiv preprint arXiv:1809.07599*, 2018.