Towards Large-Scale Graph-Based Traffic Forecasting: A Data-Driven Network Partitioning Approach

Chenhan Zhang, *Student Member, IEEE,* Shuyu Zhang, Xiexin Zou, Shui Yu, *Senior Member, IEEE*, and James J.Q. Yu, *Senior Member, IEEE*

***Abstract*—Network partitioning is recognized as an effective auxiliary approach for solving transportation tasks on large-scale traffic networks in a domain-decomposition manner. Most of the existing related partitioning algorithms are explicitly designed to traffic management problems and merely focus on the implied topology of the networks. In this paper, towards the practical problems that happened to traffic forecasting tasks, we propose a network-partitioning-based domain-decomposition framework to improve GCN-based predictors' performance on large-scale transportation networks. Particularly, we devise a data-driven network-partitioning approach, namely, Speed-Matching-Partitioning, which employs not only the topological features but also the traffic speed observations of traffic networks for partitioning. Additionally, we propose a data-parallel training strategy that feeds partitioned sub-networks into independent predictors for parallel training. The proposed approach is tested by comprehensive case studies on three real-world datasets to evaluate its effectiveness. The results indicate that the proposed approach can help improve GCN-based predictors' accuracy and training efficiency on both small and relatively large traffic datasets. Furthermore, we investigate the model sensitivity to the selection of graph representations and framework parameters, and the learning efficiency of the data-parallel training strategy.**

***Index Terms*—Network partitioning, traffic forecasting, distributed computation, graph neural network, domain decomposition.**

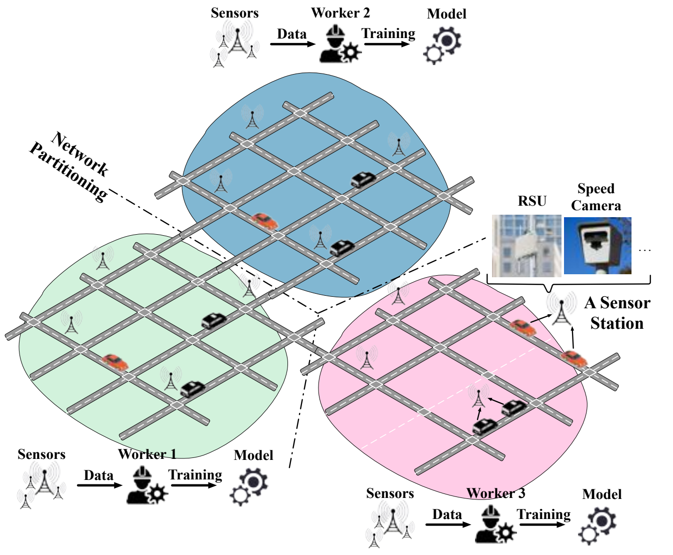


Fig. 1. An illustration of network partitioning-based task on traffic network with a crowdsourcing solution.

I. Introduction

Intelligent transportation systems (ITS) is among the key cyber-physical systems constituting one of the principal dimensions of smart cities [1]. Effective ITS can produce massive urban transport information. Recent years have witnessed a deal of research effort on traffic control optimization and vehicle management to solve various traffic problems, e.g., infrastructure allocation, driver privacy, and traffic congestion [2]–[4]. To gain better knowledge and control on the traffic, some of these ITS applications rely heavily on accurate traffic forecasting (TF) [5]–[7].

In the traffic domain, there exists a wide range of IoT sensors that can collect traffic data, e.g., radar-driven roadside units (RSU) and automatic number-plate recognition-empowered cameras. The development of the Industrial Internet of Things (IIoT) enables such reliable IoT devices to be deployed in transportation networks on a large scale to collect massive traffic data.

This trend makes deep learning-based TF approaches increasingly popular in ITS research [8]. Recently, Graph deep learning (GDL)-based methods have been seen in the emerging trend of handling TF problems [9]. By employing GDL methods, researchers can process the traffic data on graphs, which breaks through the restrictions of traditional non-spatial representation of traffic data. Graph Convolutional Networks (GCN) approach is one of the most important developments of GDL, which has demonstrated practical feasibility for TF [10].

Most of the research in this domain (i.e., GCN-based approaches for TF) focus on optimizing the predictors to improve the prediction performance, where a plethora of achievements have been made [11]. Although GCN-based approaches have been verified to be effective in handling TF, some potential

Corresponding author: James J.Q. Yu (email: yujq3@sustech.edu.cn) and Shui Yu (email: shui.yu@uts.edu.au).

This work is supported by the Stable Support Plan Program of Shenzhen Natural Science Fund No. 20200925155105002, by the General Program of Guangdong Basic and Applied Basic Research Foundation No. 2019A1515011032, by the Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation No. 2020B121201001, and by Australia ARC DP200101374 and LP190100676.

Chenhan Zhang and Shui Yu are with the Faculty of Engineering and Information Technology, University of Technology Sydney, Sydney, Australia. Shuyu Zhang and James J.Q. Yu are with the Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China. Xiexin Zou is with the Department of Electrical Engineering, The Hong Kong Polytechnic University.

issues also exist. One of them is the practicality of GCN-based models on large-scale transportation networks (i.e., number of transportation nodes greater than 1000 [12]). Unlike other graph structures (e.g., social networks) where nodes can correlate without considering their physical distances (a.k.a. Euclidean space distances), the physical position of nodes in the traffic network has a significant implication on their interaction.The primitive GCN-based predictors' performance is impeded by this condition [13]. From the view of a single node in a large-scale traffic network, the GCN operation has to use the interaction with some nodes that have an overly long distance from it. From the entire network's view, GCN on such a large-scale traffic network may involve massive irrelevant spatial information, which degenerates the final prediction performance.

To alleviate the issues, some studies focus on developing dynamic and adaptive adjacency matrix by learning the latent spatial dependency from the traffic data in node embedding [14], [15]. Others attempt to pre-process the traffic networks to make GCN-based approaches more efficient [16]. In many large-scale engineering research contexts, Domain-Decomposition (DD) has been seen as an effective approach that can develop a solution within a reasonable time [17]. Inspired by its idea, one of the strategies for pre-processing traffic networks is to use network partitioning methods to divide the entire traffic network into a group of partitions to transform the original problem into multiple parallel sub-problems [18]. Especially in recent years, the crowdsourcing scheme has been widely-adopted in IIoT, which makes DD become a quite feasible solution. In a crowdsourcing scenario, there exist multiple workers, and each worker is assigned a sub-problem (e.g., training a model with the data of a sub-network). The workers can process the sub-problem in parallel, which can effectively accelerate the progress (See Fig. 1 for an example).

Most of the existing network-partitioning approaches for transportation networks are based on the graph theory that they partition a graph considering only the topological features [17], [19], [20]. Meanwhile, the majority of them focus on traffic management optimization measured by macroscopic fundamental diagram [20]–[22]. There is not a specific data-driven network partitioning approach for assisting TF tasks. Transportation datasets generally incorporate not only topological features but also other information (e.g., speed observations, congestion index). Intuitively, leveraging these data into network-partitioning approaches can develop better partitioning results, serving GCN-based predictors to improve the prediction performance.

In this paper, to demonstrate the above idea, we propose a novel traffic data-driven network partitioning approach, i.e., Speed-Matching-Partitioning (SMP), as a domain-decomposition solution to assist TF on large-scale traffic networks. Different from the existing strategies, SMP partitions the traffic network by merging traffic data (i.e., speed value) and network topology features. Furthermore, in terms of model training, we adopt a data-parallel training framework to improve the overall training efficiency. It is worth noting that the proposed approach is orthogonal to the aforementioned dynamic and adaptive adjacency matrix-based approaches [14], [15]; they can be combined to achieve better performance. To investigate whether the proposed network partitioning is capable of boosting the TF performance of GCN-based models, we conduct a series of rigorous case studies of the proposed approach on three real-world datasets. Thanks to the traffic-data-driven network partitioning approach and the data-parallel training strategy, our proposed approach helps GCN-based predictors develop more accurate predictions while improving the training efficiency.

TABLE I
SUMMARY OF ABBREVIATIONS (IN OCCURRENCE ORDER)

Abbreviation	**Full Name**
ITS	Intelligent transportation system.
TF	Traffic forecasting.
IIoT	Industrial Internet of Things.
RSU	Roadside unit.
GDL	Graph deep learning.
GCN	Graph convolutional network.
DD	Domain-decomposition.
SMP	Speed-Matching-Partitioning (our approach).
MGP	Multilevel graph partitioning.
DBR	Distance-based rule (see Definition 2).
CBR	Connection-based rule (see Definition 3).
EBC	Edge-betweenness centrality.
SHEM	Sorted heavy-edge matching.
SV	Speed-value.

The main contribution of this paper is as follows:

- A novel Speed-Matching-Partitioning approach serving the domain-decomposition is proposed. Different from most of the existing partitioning algorithms that only consider the topology of traffic networks [17], [19], [20], towards the TF tasks, the proposed one involves the speed observations in the partitioning decision. This paper is among the pioneering studies of using traffic data characteristics to partition the traffic network.
- A network-partitioning-based domain-decomposition framework is proposed for TF on large-scale traffic networks. This framework can be trained in parallel. The distributed computing nature of this framework promises it a practical solution for crowdsourcing TF in IIoT.
- A series of comprehensive case studies are performed to verify the approach performance, seek the best structure of the proposed approach, and investigate the influence of graph representation. The results also reveal the significance of the constituting components in the proposed approach.

The remainder of this paper is organized as follows. In Section II, the background of traffic forecasting and network-partitioning research is presented. Section III gives some preceding definitions of this work. Section IV elaborates on the proposed domain-decomposition framework and network-partitioning approach. We perform a series of case studies in Section V to demonstrate their efficacy. Finally, this paper is concluded in Section VI with a summary of potential future studies.

A summary of abbreviations involved in this paper is presented in Table I.

II. Related Work

A. GCN-based Traffic Forecasting

To exploit the spatial correlation of traffic data, ITS researchers previously introduced Convolutional Neural Networks (CNN) into the TF tasks. For example, Ma *et al.* [23] proposed an image-based approach that represents the traffic networks as images and leverages CNN to learn the spatial dependency. Traditional CNNs are restricted to only process grid-like spatial structures such as images. However, data are often sampled in non-Euclidean spaces, such as graphs in the traffic context. To address this issue, Graph Deep Learning (GDL) is receiving attention from the community [24], [25]. Graph Convolutional Networks (GCN) are among the recent machine learning developments that generalize CNN to graph domains [26]. For TF problems, GCN is widely adopted to handle various tasks by treating traffic networks as graphs that can richly take advantage of spatial traffic information [13], [27]. Zhao *et al.* [28] proposed a model incorporating GCN and Gated Recurrent Unit (GRU) to exploit spatial and temporal features, respectively. Yu *et al.* [13] proposed a Spatio-Temporal Graph Convolutional Network (STGCN), which adopts convolutional structures on both spatial and time axis, which empowers faster model training speed and convergences. Besides, STGCN employs fewer parameters to achieve better scalability. Inheriting the notion of GCN-based spatial-temporal modeling, Wu *et al.* [15] integrated dilated casual convolution in their GCN model, which can extract spatial dependencies with a larger and more flexible receptive field in traffic networks. Moreover, this study constructed a self-adaptive adjacency matrix instead of a fixed adjacency matrix to make latent spatial correlations preserved. Similar ideas can be found in some other studies. The authors in [29] use the attention mechanism to dynamically weight the adjacency matrix. Guo *et al.* [14] adopted Fast-GCN [30] as the GCN model and devised dynamic Laplace matrices to extract spatial-temporal features from traffic data.

B. Network Partitioning on Traffic Networks

Research on traffic networks has been developed in the past decades for resolving traffic assignment problems or optimizing traffic management systems, such as predicting link flows [31] and route selection [32]. The main two branches are vehicle network-based approaches and road network-based approaches. Vehicle network-based approaches treat vehicles as entities to construct the network [33], [34], while road network-based approaches regard roads as entities to build the network [35], [36]. Particularly, our work falls into the realm of road network-based approaches.

No matter in which branch, network partitioning is considered a high-efficiency and time-saving approach to solving the problem due to its distributed nature. Most of the existing traffic network partitioning approaches adopt graph partitioning algorithms, which are based on graph theory and regard network partitioning as an NP-hard problem [21].

Heuristic algorithms are among the earliest graph partitioning algorithms [37], the best known being the Kernighan-Lin algorithm that can run in $O(n^3)$ time complexity [38]. Following subsequent variants of Kernighan-Lin algorithm, Fiduccia-Mattheyses further achieve a linear-time heuristic that runs faster in [39]. Other algorithms adopt different strategies to graph partition, instances of which incorporate probabilistic implementations (e.g., simulated annealing [40] and genetic algorithms [41]).

Multilevel Graph Partitioning (MGP) is another family of graph partitioning algorithms. Compared to simple heuristic algorithms that may converge to a locally optimal solution, MGP methods aim to seek globally optimal solutions that can yield higher-quality partitioning results with a small computational cost. [42] and [43] are recognized as the earliest MGP methods. Following much of the original structure of the MGP heuristic in [43], Metis is proposed to find good partitions at a faster speed [44]. The Metis software package is widely adopted in various research thanks to its various merits. Furthermore, Sanders *et al.* proposed a group of MGP methods known as Kahip, which achieve a series of good results in related contests [40], [45].

MGP algorithms demonstrate great applicability to traffic network partitioning. For example, Metis incorporates a structure that can well process large non-euclidean graphs, including traffic networks. Besides, a few methods combining the critical ideas of MGP heuristics have also shown good traffic network partitioning results [46], [47]. Few of them are based on the data generated on the traffic networks. Furthermore, [18] indicates that the TF accuracy improvement developed by these "topological-only" network-partitioning approaches is not significant. These findings motivate us to develop a data-driven network partitioning algorithm as a DD approach for improving TF accuracy.

Additionally, it is worth noting that while one common goal of our approach and the prevailing dynamic graph-empowered GCN-based predictors (e.g., the aforementioned studies [14], [15], [29]) is to solve the GCN's learning bottleneck on large-scale traffic graphs, the notions are different. The proposed approach is auxiliary, which in essence serves as a framework to unleash the GCN's power on a relatively smaller sub-graph by network partitioning and distributed training. In theory, these dynamic graph-empowered GCN-based predictors can be integrated with the proposed approach to developing better performance.

III. Preliminary

In this section, we first define the problem of traffic forecasting on graphs. Subsequently, two graph construction methods for traffic networks, which are investigated in this paper, are introduced. Then, we give a brief introduction of the GCN-based traffic predictor and a basic working principle of graph convolution.

A. Traffic Forecasting Problem

Traffic forecasting on traffic networks aims to predict the future traffic states of all road segments using previously observed traffic data.

Definition 1 Traffic forecasting on graphs. *We represent the traffic network as a weighted directed graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, *where $\mathcal{V}$ is a finite set of nodes that $|\mathcal{V}| = \mathcal{N}$, $\mathcal{E}$ is the set of edges. Finally, this problem can be formulated as*

$$[X_{t-\mathcal{T}+1}, ..., X_t; \mathcal{E}; \mathcal{G}] \xrightarrow{f(\cdot)} [\hat{X}_{t+1}, ..., \hat{X}_{t+\mathcal{H}}], \tag{1}$$

where X_t is the observed historical set of traffic data sampled from $\mathcal{N}$ sensor stations at time t, and $\hat{X}_t$ denotes the predicted set; $f(\cdot)$ is the prediction function of the model; $\mathcal{T}$ and $\mathcal{H}$ are the lengths of the past and future time window, respectively.

B. Graph Representations of Traffic Networks

In this study, traffic networks are represented as road graphs to define the spatial structure of traffic data. In particular, we regard the road segments where each contains a measuring sensor station as the graph nodes in this representation. Generally, an original road graph possesses no edge information. In this paper, we focus on the GCN performance with static graph structure input. To construct the edges that correlate the nodes of the graph, we introduce two common methods to build edges as follows.

Definition 2 Distance-based rule (DBR). *In this method, an edge is constructed depending on the Euclidean space distances between two nodes [13], [27]. We consider two nodes to correlate if they are close enough. Based on Gaussian kernel method, the corresponding adjacency matrix $\mathcal{A}$ incorporating edge information is constructed as $\phi_{ij} = 1$ if $i \neq j$ and $\exp\left(-\frac{\mathrm{d}(i,j)}{q^2}\right) \geqslant \mu$, otherwise 0, where $\phi_{ij} \in \mathcal{A}$ represents the connectivity between nodes i and j, $\mathrm{d}(i,j)$ denotes the Euclidean space distance between nodes i and j; μ and q denote user-defined thresholds that control the sparsity of the matrix and are empirically set to 10 and 0.5 as did in [13], [27], respectively. Particularly, $\phi_{ij} = 1$ indicates an edge between nodes i and j, while $\phi_{ij} = 0$ indicates no inter-connecting edges.*

Definition 3 Connection-based rule (CBR). *This method defines an edge following the actual connection of different road segments [48]. A directed edge is constructed if there exists a connection between two road segments. An example of this graph representation is shown in Fig. 2, where eight road segments are labeled from 1 to 8 and are regarded as the nodes. For node 1 (i.e., road segment 1), there are three nodes, i.e., 2, 4, 6, connecting to it directly. Thereby, there are edges (1, 2), (1, 4), (1, 6), respectively. Additionally, we define the two directions of bidirectional roads as two nodes (e.g., nodes 2 and 3).*

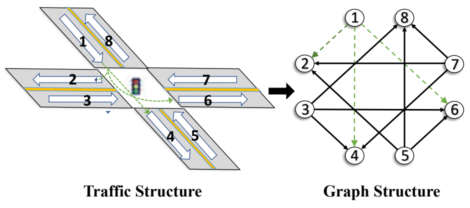


Fig. 2. Connection-based rule of edge construction.

C. GCN-based Traffic Predictor

GCN-based TF models are recognized as state-of-the-art due to their effectiveness in traffic spatial information extraction. In this subsection, we take Spatio-Temporal Graph Convolutional Network (STGCN) [49] as an example to briefly introduce the spatial extracting principles of GCN-based models. STGCN model incorporates several spatial data processing modules to exploit spatial dependency. The spatial module performs a graph convolutional operation, which follows the spectral GCN method that convolves in the spectral domains [26].

Definition 4 Graph convolution (spectral domain-based). *[?] To process the graph data in the spectral domain, the normalized graph Laplacian matrix is first computed given the adjacency matrix $\mathcal{A}$ as*

$$\mathcal{L} = I_N - D^{-\frac{1}{2}} \mathcal{A} D^{-\frac{1}{2}}, \tag{2}$$

where $I_N \in \mathbb{R}^{N*N}$ is the identity matrix and $D = diag\left(\Sigma_j \phi_{ij}\right) \in \mathbb{R}^{N*N}$ is the diagonal degree matrix. Then, $\mathcal{L}$ is decomposed as

$$\mathcal{L} = U \Lambda U^T, \tag{3}$$

where $U \in \mathbb{R}^{N*N}$ is the matrix of eigenvectors of $\mathcal{L}$; $\Lambda \in \mathbb{R}^{N*N}$ is the diagonal matrix of κ with $\Lambda = diag(\kappa)$, and κ is the eigenvalues of $\mathcal{L}$ in descending order. Subsequently, a convolution operation is performed in the spectral domain of the graph and can be formulated as

$$g_\theta * X = g_\theta\left(U \Lambda U^T\right) X = U g_\theta(\Lambda) U^T X, \tag{4}$$

where X is the input data, $*$ denotes the graph convolutional operator, and g_θ is the kernel with a group of convolution parameters represented by $\theta \in \mathbb{R}^N$. Thus, an updated feature of the input X can be computed by the multiplication between g_θ and $U^T X$. Consequently, the spatial correlations among different nodes in the graph can be learned and stored in the new features. The graph convolutional operation can be finally formalized as

$$X' = \sigma\left(\widetilde{D}^{-\frac{1}{2}} \widetilde{\mathcal{A}} \widetilde{D}^{-\frac{1}{2}} X \theta_g\right), \tag{5}$$

where X' is the update which contains updated nodal embeddings; $\widetilde{\mathcal{A}} = \mathcal{A} + I_N$ is the adjacency matrix with added self-connections, $\widetilde{D}$ is the diagonal degree matrix of $\widetilde{\mathcal{A}}$, θ_g denotes all the shared parameters in this operation, and $\sigma(\cdot)$ is the sigmoid function.

IV. Network Partitioning-based Domain-Decomposition Framework

In this work, we propose a network partitioning-based domain-decomposition framework along with a novel network partitioning approach, Speed-Matching-Partitioning (SMP), for GCN-based predictors on large-scale traffic networks. In this section, we first discuss the problem of GCN's performance on large-scale traffic networks, which motivates us to develop our method. Then, we give an overview of the proposed framework, including its basic working pipeline. Subsequently, we elaborate on the algorithm of SMP, which is the most important component of the proposed framework.

A. Performance of GCN on Large-scale Networks

While GCN-based predictors like STGCN [13] have been widely verified to be effective in handling TF, there still exist some limitations. A large-scale urban traffic network may have thousands of road segments distributed in an intricate pattern. GCN propagate embeddings using the interaction among nodes in the graph, which makes the learning procedure quite challenging when the graphs are large and complex. In this subsection, we discuss two types of degeneration of GCN performance on large-scale traffic networks.

1) Learning Capacity Degeneration of GCN on Large-scale Networks: The learning capacity of GCN may degenerate when transportation networks go large. Unlike other graph structures (e.g., social networks) where nodes can correlate without considering their physical distances (a.k.a. Euclidean space distances), the physical position of nodes in the traffic network has a significant implication on their interaction. However, the graph construction methods like the introduced DBR and CBR inevitably connect some nodes which exactly have weak latent spatial correlations. From the view of a single node in a large-scale traffic network, the GCN operation on the node has to use the interaction with such nodes, which have weak latent spatial correlation (as illustrated in the top left corner of Fig. 3). This cannot effectively contribute to the spatial information learning of the node and, sometimes, can cause noises that determine the quality of learned spatial information. Moreover, if there is more than one GCN layer in the model, accumulated interactions will further enlarge this negative impact [50]. Consequently, the final prediction performance of the GCN-based predictors will be depreciated.

2) Training Efficiency Degeneration of GCN on Large-scale Networks: Large-scale networks also result in low training efficiency of GCN. The loss term in GCN cannot be precisely decomposed into individual terms on each sample, which depends on a mass of other nodes [26]. Due to their inter-dependency, GCN training imposes a significant memory footprint since back-propagation needs to store all the embeddings from the computational graph to GPU memory. Furthermore, GCN yields a time complexity of $O\left(L\|\mathcal{A}\|_0 F + L\mathcal{N}F^2\right)$ [51], where L denotes the number of layers, $\mathcal{N}$ denotes the number of nodes in the traffic network, $\|\mathcal{A}\|_0$ denotes the number of nonzeros in the adjacency matrix, and F is the number of hidden features in the neural networks. Obviously, when the graph is large and sparse, the time consumption of training GCN will significantly increase.

B. Framework Overview

In this work, to improve the efficiency of GCN-based models on large-scale traffic networks and thus improve the TF performance, we introduce a domain-decomposition-based framework, as shown in Fig. 3. Specifically, we first leverage network partitioning approaches to decompose a large urban network into S smaller sub-networks. We assume that this process can improve the overall efficiency of GCN. Additionally, a data-parallel training scheme is adopted to train the S partitions separately and simultaneously on the corresponding number of independent computing devices. This data-parallel training scheme can achieve a $\eta_S = t_1/t_S$ times speedup in ideal conditions, where t_1 denotes the time to run the model on a single compute node, and t_S denotes the time to run the model on S parallel compute nodes.

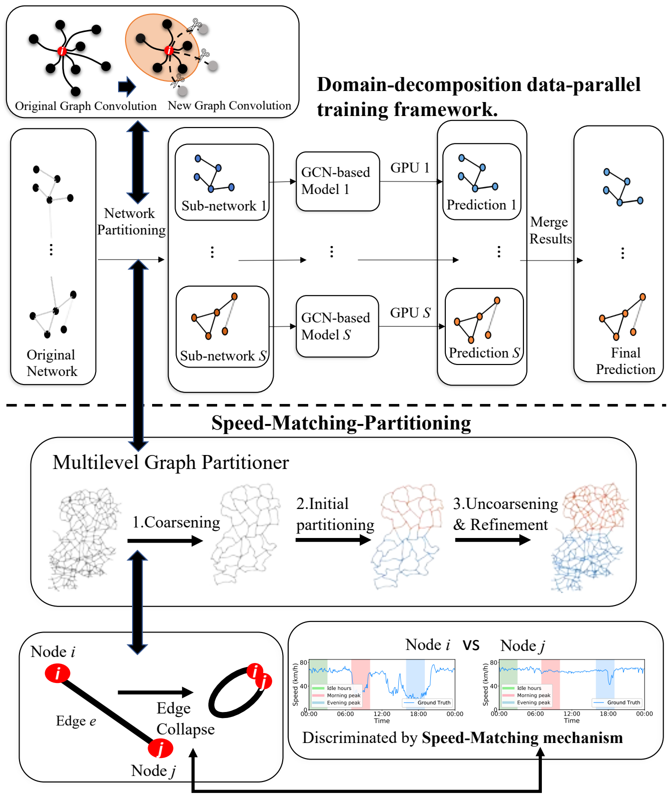


Fig. 3. The schematic of the proposed approach.

C. Speed-Matching-Partitioning Approach

Before our elaboration, we first introduce the graph-theoretic terms involved in the adopted approaches.

Definition 5 Cut. *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, A cut $C = (\mathcal{V}_1, \mathcal{V}_2)$ is a partition of $\mathcal{V}$ into two subsets $\mathcal{V}_1$ and $\mathcal{V}_2$, whose removal makes a graph disconnected. We have a cut-set of C, which is the set of edges satisfying $\{(i,j) \in \mathcal{E} | i \in \mathcal{V}_1 \wedge j \in \mathcal{V}_2\}$, i.e., one endvertex in $\mathcal{V}_1$ and the other endvertex in $\mathcal{V}_2$.*

Definition 6 Betweenness centrality. *Betweenness centrality is a measure of centrality in a graph based on shortest paths [52]. Edge-Betweenness Centrality (EBC) is defined as the number of the shortest paths that go through an edge in a graph, which is defined as*

$$\mathrm{E}(e) = \sum_{i \neq j \in \mathcal{V}, e \in \mathcal{E}} \frac{\delta(i, j|e)}{\delta(i,j)}, \tag{6}$$

where $\delta(i,j)$ denotes the number of different shortest paths going through nodes i and j—(i,j)-paths, and $\delta(i,j|e)$ denotes the number of shortest (i,j)-paths which contain e as an inner edge.

Definition 7 Edge collapse. *Edge collapse (a.k.a. edge contraction) is an operation that eliminates an edge from a graph and concurrently merge the two vertices it previously*

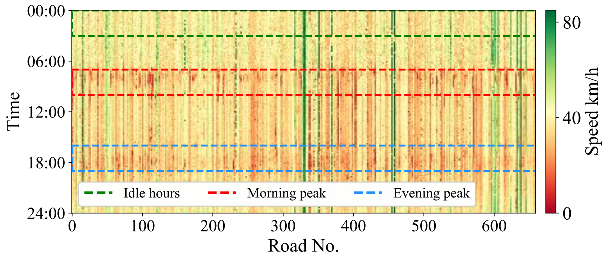


Fig. 4. The speed observations in peak hours and idle hours.

connected [53]. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ incorporating edge $e = (i, j), i \neq j$, the collapse of e deduces a new graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ with a new node k where $\mathcal{V}' = (\mathcal{V}' \backslash \{i, j\} \cup \{k\})$.

1) EBC Pre-processing-based MGP Algorithm: MGP algorithm involves three phases: (1) *coarsening*, (2) *initial partitioning*, and (3) *uncoarsening and refinement*. In the first two phases, it is crucial to find suitable candidates to be cut-edges for partitioning. Usually, a road network can be separated into different areas by natural and artificial barriers (e.g., rivers, mountains, and urban boundaries). The sparsely built roads can serve as good cut-edge candidates since they naturally overcome these barriers, namely, natural cuts[1]. Natural cuts are recognized with high EBCs [54]. Not only natural cuts but also some edges that possess high EBC can also be regarded as good cut-edges. We refer to these cut-edges with high EBCs as High EBC cuts (HEBC-cuts). While EBC might be a useful reference for providing us a promising pre-processing method for attaining good cut-edges, it is computationally intractable [52]. To address this problem, following [55], we approximate EBC by computing the shortest paths for only a small sample of node pairings $\mathcal{V}^*$ with $\mathcal{V}^* \in \mathcal{V}$ instead of the complete EBC, through comparing the shortest paths for all nodes in a graph pairwise. This approximation can be formulated as

$$\mathrm{E}_{appr}(e) = \frac{1}{\mathcal{N}(\mathcal{N}-1)} \sum_{i \neq j \in \mathcal{V}^*, e \in \mathcal{E}} \frac{\delta(i, j|e)}{\delta(i, j)}, \tag{7}$$

where $\mathrm{E}_{appr}(e)$ denotes the approximate EBC of edge e and $\mathcal{N} = |\mathcal{V}|$. Additionally, we apply a random strategy when selecting the nodes to pairwise compare, which prevents the selected nodes from being centered in a narrow area of the graph.

In this work, Metis is employed as the default MGP partitioner, which applies sorted heavy-edge matching (SHEM) algorithm [56] in coarsening and initial partitioning phases. A notion of our design is that SMP can be transplanted to different partitioners incorporating SHEM. In particular, SHEM assigns discretized weights to the edges according to their EBCs. On the one hand, we assign high edge weights to edges with a low EBC and low edge weights to edges with a high EBC, and the edges with a low EBC are preferentially included in the matching, which will be *collapsed* consequently. On the other hand, we discretize the assigned weights to avoid exaggerated EBCs when the nodes sample is small. This is because two equivalently significant HEB-cuts may be assigned magnitude apart weights. In this way, all edges distributed in a specific range of approximate EBC are assigned the same weight. The EBCs of edges in a road network always follow a power-law distribution, which also happens to approximate EBC. Thereby, a mapping scheme is proposed and can be formulated as

$$w_e(e_{ij}) = \perp\!\perp\!\perp\left(\ln\left(\frac{1}{\mathrm{E}_{appr}(e_{ij})}\right)\right), \tag{8}$$

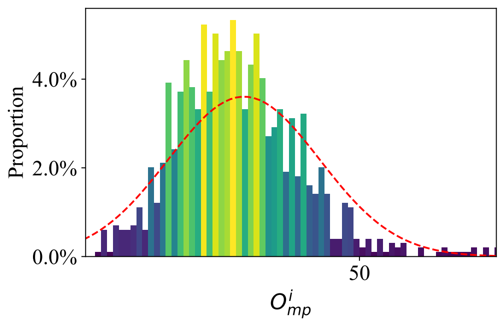
where e_{ij} denotes the edge that connects nodes i and j; $w_e(e_{ij})$ is the weight assigned to edge e; $\perp\!\perp\!\perp(\cdot)$ denotes the process of discretization by rounding, to which the SHEM algorithm is sensitive [57]. Consequently, it is easy to differentiate the potential cut-edges since HEB-cuts are more likely to survive the coarsening stage.

Algorithm 1: Speed-Matching-Partitioning approach.

```
Input: Original graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$; Historical traffic
       observations (traffic data); The number of
       partitions, $S$; The base factor in formula (10),
       $b$; The weights assigned to different time
       periods, $\lambda_{mp}$, $\lambda_{ep}$, and $\lambda_{ih}$; A SHEM-enabled
       MGP.
Output: A group of partitions, $\{\mathcal{G}^*_1, ..., \mathcal{G}^*_S\}$.
1  $\mathcal{A} \leftarrow$ construct adjacency matrix of $\mathcal{G}$ via Def. 2 or 3
2  foreach each $e \in \mathcal{E}$ do
3      $\mathrm{E}_{appr}(e)\theta_i^* \leftarrow$ approximate the EBC of edge $e$ via
        Eq. (7)
4      $\omega_e(e)\theta_i^* \leftarrow$ compute the EBC-based weight via Eq.
        (8)
5  foreach each $i \in \mathcal{V}$ do
6      foreach each $j \in \mathcal{V} \mid i \neq j$ do
7          if $\phi_{ij} == 1 \mid \phi_{ij} \in \mathcal{A}$ then
8              $\vartheta_i, \vartheta_j \leftarrow$ calculate the SVs of node $i$ and
                node $j$ based on their speed observations
                via Eq. (9) and (10)
9              $\varphi_{ij} \leftarrow$ calculate the magnifying factor for
                edge $e_{ij}$ via Eq. (11)
10             $w_e^*(e_{ij}) \leftarrow$ compute the magnified weight
                via Eq. (11)
11 Configure all magnified weights $w_e^*$ to the pre-defined
    MGP partitioner
12 $\{\mathcal{G}^*_1, ..., \mathcal{G}^*_S\} \leftarrow$ use the MGP partitioner to divide $\mathcal{G}$
    into $S$ partitions
```

2) Speed-Matching Mechanism: In this paper, we propose a Speed-Matching mechanism and incorporate it into the aforementioned EBC pre-processing-based MGP algorithm, which forms the final Speed-Matching-Partitioning approach. Conventional graph partitioning approaches identify the cut-edges by only considering the topological relationships between nodes. While these approaches can develop satisfactory partitioning results, for the TF task as defined in this work, it has been verified that GCN-based predictors cannot obtain distinctively better performance on these developed partitions [18]. To overcome this issue, we introduce a speed value

[1]We define the connections between road segments as edges in this work and the cut-edges are therefore these road connections.



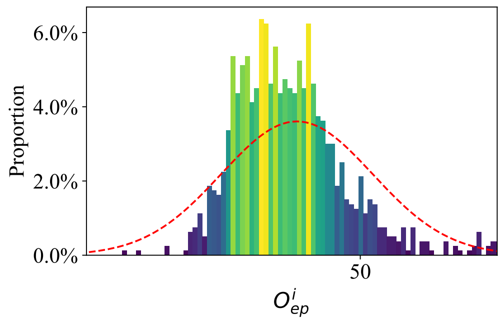


Fig. 5. Probability distribution functions of O^i in different time quantums. The x-axes indicate the values of O^i_{mp} and O^i_{mp}. The y-axes indicate the proportion of different values of O^i_{mp} and O^i_{mp}.

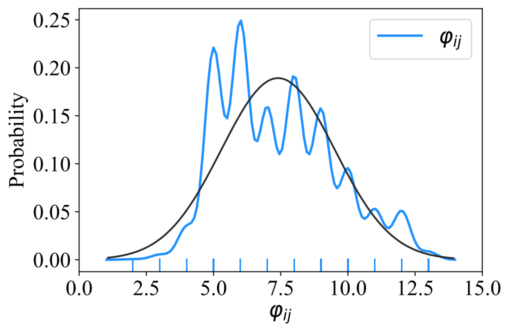


Fig. 6. Distribution of φ_{ij} developed by Eq. (11) with γ derived by Monte Carlo Approach. We can see the values of φ_{ij} are bound in $[2, 13]$ and close to normal distribution.

matching mechanism into cut-edges identified in the coarsening step of MGP, as shown in Fig. 3.

To "match" the speed values of two nodes of an edge (i.e., two road segments), we introduce a measured value, **Speed-Value (SV)**, represented by ϑ. We commence by calculating the SVs. It is known that the observed traffic speeds of a road segment vary over time. Therefore, it is not representative to arbitrarily select the average value of the speed observations within a day as the speed value. Thus, three representative periods regarding distinct speed observation are sampled, i.e., morning peak (7:00–10:00), evening peak (16:00–19:00), and idle hours (0:00–3:00), to calculate a weighted average value as SV. As shown in Fig. 4, the observations in the peak periods and the idle hours can reflect the congested speed and the maximum speed, respectively. The average speed at node i is represented by O^i, and be calculated by

$$O^i = \frac{1}{QP} \sum_{z=1}^{Q} \sum_{u=1}^{P} x^i_{z,u}, \tag{9}$$

where Q is the number of sampled days in the adopted dataset; P is the number of timestamps in the dedicated period; z and u are the index numbers for Q and P, respectively; x denotes the speed observation, and $x^i_{z,u}$ represents the speed observation at timestamp u from day z. Subsequently, the O^i of morning peak, evening peak and idle hours can be obtained, respectively, which are denoted as O^i_{mp}, O^i_{ep}, and O^i_{ih}. As shown in Fig. 5, there exists difference of the distribution between different O_i, which contribute towards obtaining dispersed ϑ. Afterwards, we calculate the SV using linear weighted aggregative method discretization, which can be formulated as

$$\vartheta_i = \perp\!\perp\!\perp(\log_b(\frac{1}{\underbrace{\lambda_{mp} O^i_{mp}}_{\text{morning peak}} + \underbrace{\lambda_{ep} O^i_{ep}}_{\text{evening peak}} + \underbrace{\lambda_{ih} O^i_{ih}}_{\text{idle hours}}})), \tag{10}$$

where ϑ_i denotes the calculated SV of node i; λ_{mp}, λ_{ep}, and λ_{ih} are the weights for O^i_{mp}, O^i_{ep}, and O^i_{ih}, respectively; b is the base factor which can control the level of discretization. Note that the weights, i.e., λ_{mp}, λ_{ep}, λ_{ih} and b are user-controlled parameters, and we set them to 0.4, 0.4 and 0.2 by default. Related case studies about the selection of λ_{mp}, λ_{ep}, λ_{ih}, and b will be shown in Section V-C. Additionally, we know that the traffic observations vary from time and the change pattern of traffic states vary from different areas. In practice, Eq. (9) and (10) can be executed in adaptive modes for capturing the dynamics (e.g., dynamical update of input traffic speed observations). In this way, we can finally develop dynamic-aware partitions to serve downstream tasks that demand real-time performance (cf. online learning).

Subsequently, we apply a matching mechanism in the process of edge collapse. From the previous chapter, it is clear that the graph is coarsened by collapsing the edges which have low EBCs. Based on this, we further collapse the edges whose two nodes have similar SVs. This can be physically explained that such two nodes (i.e., road segments) with similar SVs are more likely to be in the same flow in the road network, and their connecting edge should survive in the partitioning. To achieve this, we magnify the edge weight w_e introduced in (9) by the difference of SVs between two endpoints of edge e, i.e., $|\vartheta_i - \vartheta_j|$. This process is designed as

$$\begin{aligned}\varphi_{ij} = \log_{\vartheta_r+\gamma_1}\left(\frac{\vartheta_r+\gamma_2}{(|\vartheta_i-\vartheta_j|)+\gamma_3}\right) + \\ \log_{\vartheta_r+\gamma_4}\left(\frac{\vartheta_r+\gamma_5}{\vartheta_r+\gamma_6}\right),\end{aligned} \tag{11}$$

$$w^*_e\left(e_{ij}\right) = w_e\left(e_{ij}\right)\varphi_{ij}, \tag{12}$$

where φ_{ij} is the magnifying factor, ϑ_r is the difference between the maximum SV and the minimum SV among nodes; the constant terms $\gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5, \gamma_6$ are used to suppress overdispersed results where we use Monte Carlo approach [58] to obtain a proper portfolio of γ (See Fig. 6 for an illustration of the results); w^*_e is the magnified weight. Then, the partitioner utilizes these magnified weights to discriminate whether an edge should be collapsed and develop the final partitioning results. For a better understanding of the audience, we present the procedures of SMP in order in Algorithm 1.

V. Case Studies

This section undertakes a group of case studies on three real-world traffic datasets to evaluate the proposed approach. We first investigate the accuracy of predicted speed by comparing the proposed approach with baselines. Then, we study how graph construction methods influence the network partitioning results and further influence the prediction accuracy. Subsequently, we examine the impact of the involved parameters of the proposed approach on model performance. Lastly, we assess the training efficiency of the proposed approach.

A. System Configuration

1) Dataset Description: In this work, three real-world network-wide traffic speed datasets are utilized to verify the proposed approach. **Nav-BJ** and **Nav-SH** are collected from NavInfo[2]. Nav-BJ contains traffic data collected from 1159 sensor stations deployed in Beijing city. Nav-SH contains data collected from 1335 sensor stations in Shanghai city. The periods of these two datasets are both from March 1st to March 31st of 2019. **PeMSD7** is a public dataset collected from Caltrans Performance Measurement System (PeMS) by 228 sensor stations in District 7 of California. The data collection period is from May 1st to June 30th of 2012 (without weekends). In all three datasets, traffic speed observations are aggregated into 5-minute interval, and Z-Score normalization is applied. Additionally, when there are missing data points, linear interpolation is adopted to recover the missing data. The training, testing, and validation sets are correspondingly generated, each of which contains 60%, 20%, and 20% of all data.

2) Compared Approaches: To assess the effectiveness of our proposed SMP approach, we compare our approach with the following baselines: (1) **HA**: Historical Average; (2) **ARIMA**: Auto-Regressive Integrated Moving Average model; (3) **GRU**: A GRU-based model; (4) **STGCN**: Original STGCN model [13]. Additionally, other network partitioning-based approaches are also introduced to verify the efficacy of our proposed graph-clustering method, including (5) **Random-STGCN**: Using the random graph-partitioning method; (6) **Metis-STGCN**: Using the naive Metis partitioner; (7) **Kahip-STGCN**: Using the naive Kahip partitioner. We demonstrate our approach as two variants i.e., **SMP-Metis-STGCN** and **SMP-Kahip-STGCN**, which use Metis and Kahip as the MGP partitioner, respectively. We use Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE) to evaluate the prediction accuracy of all approaches, which are defined as

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(X_i - \hat{X}_i\right)^2}, \tag{13a}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|X_i - \hat{X}_i\right|, \tag{13b}$$

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{X_i - \hat{X}_i}{X_i + \xi}\right| \times 100\%, \tag{13c}$$

where X_i and $\hat{X}_i$ are the ground truth and predicted value, respectively; ξ[3] is a constant which prevents the divide-by-zero issue if $X_i = 0$. We consider MAPE the most referable one among the three metrics, which accords with the common practice [48], [59].

3) Experiment Setting: Unless otherwise noted, all case studies adopt the following setting. All tests are conducted on computing servers with Intel(R) Xeon(R) E5-2620 v4 CPU and nVidia GeForce RTX 2080 Ti GPUs. For TF, the past time window is 60 minutes (12 observed data points), and they are used to forecast traffic speed in the next 15, 30, and 45 minutes. We employ Adam optimizer to train the involved neural networks (i.e., STGCN and GRU) for 100 epochs with a batch size of 50, where the learning rate is $1e^{-3}$. For STGCN, we follow the parameters setting in [13] where the set of the hidden sizes for the spatial and temporal modules is $\{1, 32, 64, 64, 32, 128\}$, and the STGCN models configured in all the related approaches follow this setting for fairness. For our proposed network partitioning approach, we set the number of partitions $S = 8$; the weights λ_{mp}, λ_{ep}, and λ_{ih} to be 0.4, 0.4, and 0.2; the base factor $b = 2$; and the rule of edge construction to be the DBR.

B. Prediction Accuracy

Tables II, III and IV demonstrate the results of our proposed approach and the aforementioned baselines on Nav-BJ, Nav-SH and PeMSD7, respectively. The best results are highlighted in bold. With simpler models compared to advanced deep learning models, conventional methods, i.e., HA, and ARIMA, have relatively large prediction errors. STGCN generally has the best performance than the other baselines, credited to its spatial feature learning capacity powered by GCN. The network-partitioning-based approaches markedly surpass the methods above. Notably, the two models of our proposed approach (i.e., SMP-Metis-STGCN and SMP-Kahip-STGCN) obtain the best results, whose MAPEs outperform original STGCN by 1.84% (15 min), 1.88% (30 min), and 1.93–2.07% (45 min) on Nav-BJ; and 1.14–1.19% (15 min), 0.81–0.82% (30 min), and 1.11–1.13% (45 min) on Nav-SH. It implies that network-partitioning-based methods are effective for STGCN to learn the spatial dependency of traffic graphs to further improve its prediction capacity. Moreover, compared with those network partitioning-based models using the naive partitioners (i.e., Metis-STGCN and Kahip-STGCN), distinguished performance improvements by our models can also be

[2]http://www.nitrafficindex.com

[3]We set $\xi = 0.01$ in the case studies

TABLE II
PERFORMANCE COMPARISON ON NAV-BJ DATASET.

Model	15 min			30 min			45 min		
	RMSE	MAE	MAPE (%)	RMSE	MAE	MAPE (%)	RMSE	MAE	MAPE (%)
HA	7.54	4.19	12.28	7.54	4.19	12.28	7.54	4.19	12.28
ARIMA	8.03	4.39	10.88	12.21	5.91	12.66	19.07	8.85	18.56
GRU	7.34	4.02	13.56	7.61	4.23	14.32	7.56	4.32	14.67
STGCN	4.53	3.10	11.13	4.84	3.28	11.84	5.02	3.40	12.30
Random-STGCN	4.21	2.87	9.70	4.47	3.07	10.65	4.63	3.18	11.19
Metis-STGCN	4.21	2.86	9.68	4.45	3.05	10.55	4.61	3.17	11.12
Kahip-STGCN	4.20	2.84	9.72	4.43	3.02	10.51	4.56	3.14	10.97
SMP-Metis-STGCN	**4.19**	**2.81**	**9.29**	**4.41**	**2.98**	**10.06**	**4.50**	**3.06**	10.37
SMP-Kahip-STGCN	**4.19**	**2.81**	**9.29**	**4.41**	**2.98**	**10.06**	**4.50**	**3.06**	**10.22**

TABLE III
PERFORMANCE COMPARISON ON NAV-SH DATASET.

Model	15 min			30 min			45 min		
	RMSE	MAE	MAPE (%)	RMSE	MAE	MAPE (%)	RMSE	MAE	MAPE (%)
HA	6.83	3.74	11.11	6.83	3.74	11.11	6.83	3.74	11.11
ARIMA	5.29	3.94	11.27	5.46	3.97	12.77	5.98	4.21	14.02
GRU	4.96	3.16	10.27	4.94	3.29	10.89	5.09	3.42	11.44
STGCN	5.35	3.22	10.69	5.21	3.29	10.89	5.09	3.42	11.43
Random-STGCN	4.48	3.02	10.05	4.70	3.18	10.70	4.80	3.25	10.92
Metis-STGCN	4.44	3.00	9.99	4.65	3.14	10.56	4.76	3.22	10.80
Kahip-STGCN	4.45	2.95	10.01	4.67	3.16	10.60	4.77	3.23	10.88
SMP-Metis-STGCN	**4.36**	**2.87**	**9.50**	**4.53**	**3.01**	**10.07**	**4.63**	**3.08**	**10.30**
SMP-Kahip-STGCN	4.39	**2.87**	9.55	4.56	**3.01**	10.08	**4.63**	**3.08**	10.32

TABLE IV
PERFORMANCE COMPARISON ON PEMSD7 DATASET.

Model	15 min			30 min			45 min		
	RMSE	MAE	MAPE (%)	RMSE	MAE	MAPE (%)	RMSE	MAE	MAPE (%)
HA	7.20	4.01	10.61	7.20	4.01	10.61	7.20	4.01	10.61
ARIMA	9.00	5.55	12.92	9.13	5.86	13.94	9.38	6.27	15.20
GRU	4.15	2.35	7.25	5.36	3.04	9.12	6.19	3.52	10.14
STGCN	3.55	2.02	4.82	4.91	2.85	7.10	5.45	3.14	7.67
Random-STGCN	3.52	2.06	4.74	4.73	2.67	6.51	5.37	3.04	7.52
Metis-STGCN	3.49	1.99	4.67	4.59	2.58	6.19	5.34	3.03	7.45
Kahip-STGCN	3.49	1.93	4.59	4.58	2.50	6.09	5.40	2.90	7.27
SMP-Metis-STGCN	3.44	1.92	4.56	4.55	**2.49**	**6.03**	5.35	**2.87**	**7.11**
SMP-Kahip-STGCN	**3.43**	**1.91**	**4.55**	**4.52**	2.50	6.07	**5.34**	2.89	7.19

TABLE V
PERFORMANCE COMPARISON ON GRAPH WAVENET.

Graph WaveNet	RMSE	MAE	MAPE (%)
Original	5.86	3.11	8.21
Metis	5.95	3.19	7.94
SMP-Metis	**5.67**	**3.08**	**7.72**

TABLE VI
PREDICTION ACCURACY (MAPE (%)) ON THE PARTITION

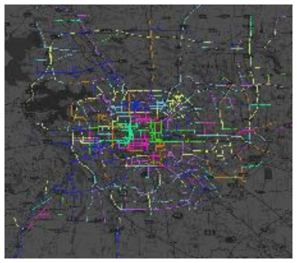
STGCN on Nav-BJ									
Partition	1	2	3	4	5	6	7	8	Total
No SMP	10.53	15.62	9.13	10.69	10.27	10.58	8.34	9.98	10.66
SMP	**10.29**	**14.19**	**8.88**	**10.32**	**10.15**	**10.16**	**8.07**	**9.60**	**10.22**

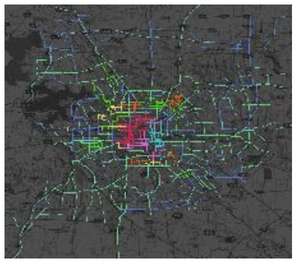
STGCN on Nav-SH									
Partition	1	2	3	4	5	6	7	8	Total
No SMP	10.17	9.73	13.95	10.03	11.00	9.72	12.58	11.35	11.06
SMP	**9.95**	**9.55**	**13.34**	**9.63**	**10.56**	**9.40**	**12.05**	**10.91**	**10.67**

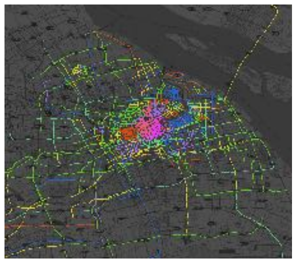
STGCN on PeMSD7									
Partition	1	2	3	4	5	6	7	8	Total
No SMP	7.72	8.03	7.70	6.27	8.98	7.27	7.40	6.65	7.53
SMP	**7.28**	**7.68**	**7.55**	**6.17**	**7.97**	**6.96**	**6.95**	**6.34**	**7.13**

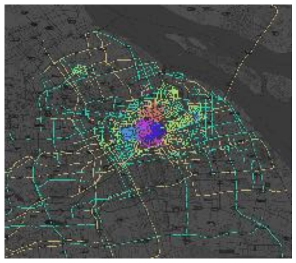
witnessed. For example, the SMP-Metis-STGCN outperforms Metis-STGCN by 0.39% (15 min), 0.49% (30 min), and 0.75% (45 min) on Nav-BJ. This is due to the proposed speed value matching mechanism enabling the network-partitioning approach to retain critical spatial correlation during partitioning to exert the capacity of GCN in spatial learning. We also observe that the performance of the road segments in the border of the sub-networks are worse than those in the center because of the lack of neighboring roads. Additionally, it can be seen that the performance differences between approaches using Metis and Kahip as the partitioners are minuscule. This can be explained by that the adoption of MGP partitioners is not the main factor in improving model performance.

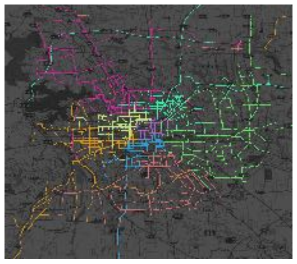
To assess the generalization ability of SMP, we also compare the prediction accuracy of Graph WaveNet [15] with/without

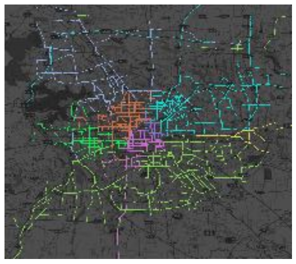

(a) SMP-Metis on Nav-BJ (distance-based).

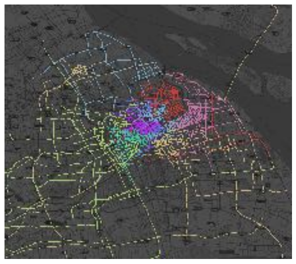

(b) SMP-Kahip on Nav-BJ (distance-based).

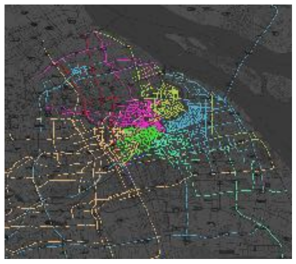

(c) SMP-Metis on Nav-SH (distance-based).

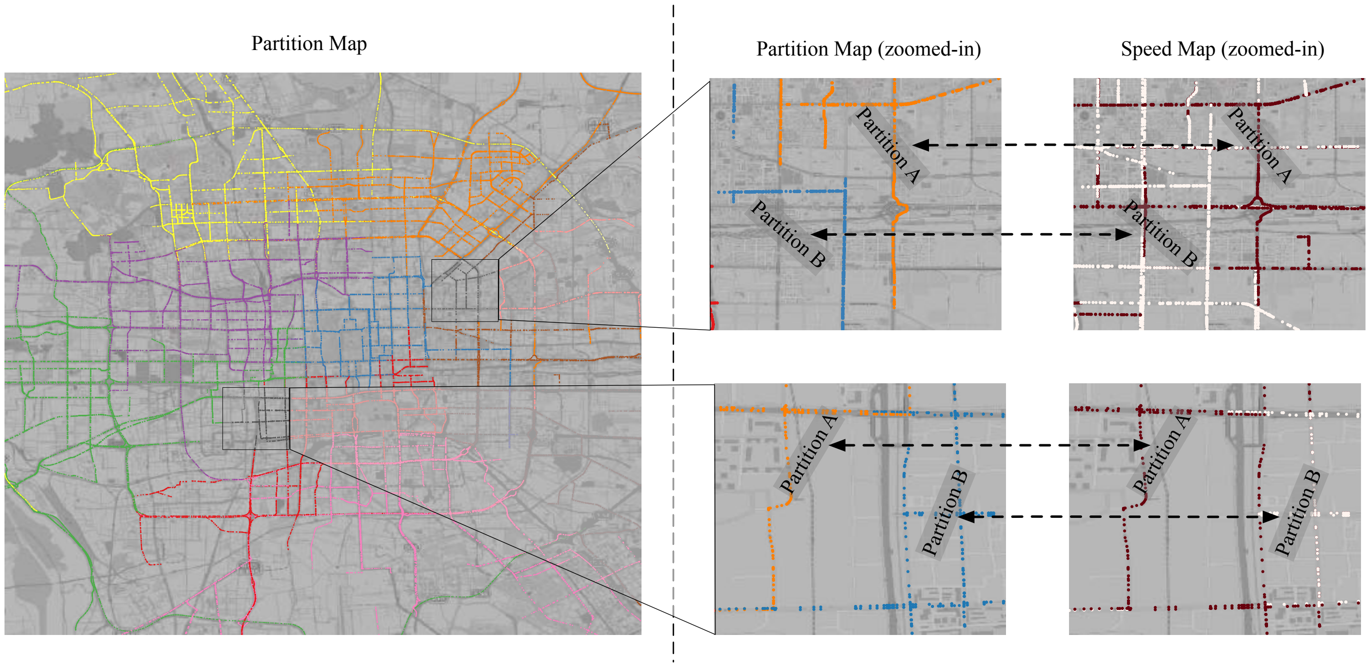

(d) SMP-Kahip on Nav-SH (distance-based).


(e) SMP-Metis on Nav-BJ (connection-based).


(f) SMP-Kahip on Nav-BJ (connection-based).


(g) SMP-Metis on Nav-SH (connection-based).


(h) SMP-Kahip on Nav-SH (connection-based).

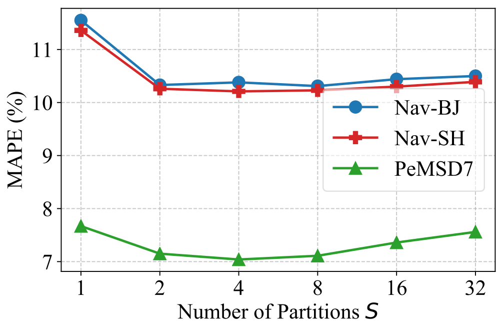


(i) Two illustration of the speed observations on the areas around the partition boundary. In the speed map, the speed observations are collected at 18:00 (peak hour) in Mar 28, 2019 on Nav-BJ, where the white dots represent the speed observations distributed in 7.7-33.5 km/h while the red dots represent the speed observations distributed in 33.5-130 km/h.

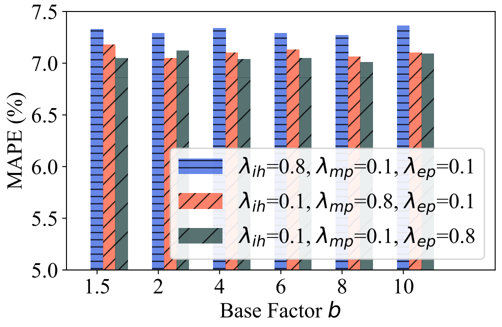
Fig. 7. Partitioning results of the proposed approach.

SMP configurations. We present the results of 45 min prediction on PeMSD7 as representative, as shown in Table V. We can find the improvement by SMP on the prediction accuracy of Graph WaveNet. It is worth noting that the diffusion convolution used in Graph WaveNet differs from the spectral graph convolution used in STGCN, as introduced in Definition 4. Therefore, we can confirm the generalization ability of SMP.

Additionally, we compare the prediction accuracy performance of STGCN under with/without SMP configurations on eight partitions. As the simulation results shown in Table VI, we can observe that on all the eight partitions, STGCN with SMP outperforms the one without SMP. This result supports our statement that the capacity of spatial-temporal GCN model can be given full play on smaller graph partitions developed by the proposed SMP. Meanwhile, we notice that the degree of improvement is different on different partitions; this question will be investigated in future work.



(a) Prediction accuracy versus Number of partitions.



(b) Prediction accuracy versus SV calculating factors.

Fig. 8. Hyperparameter sensitivity.

TABLE VII
PREDICTION ACCURACY VERSUS EDGE CONSTRUCTION METHODS.

	Method	No. of Edges	Model	Accuracy (MAPE)
Nav-BJ	DBR	4556	SMP-Metis-STGCN	10.37%
			SMP-Kahip-STGCN	10.31%
	CBR	4225	SMP-Metis-STGCN	10.50%
			SMP-Kahip-STGCN	10.51%
Nav-SH	DBR	6452	SMP-Metis-STGCN	10.30%
			SMP-Kahip-STGCN	10.32%
	CBR	5047	SMP-Metis-STGCN	10.29%
			SMP-Kahip-STGCN	10.37%

C. Selection of the Graph Construction

In Section IV, we introduce two edge construction methods to correlate the nodes, namely, Distance-based Rule (DBR) and Connection-based Rule (CBR). This subsection investigates whether the edge construction methods make a notable performance difference. We construct both the DBR and CBR topology for road segments in Nav-BJ and Nav-SH, and apply SMP to them[4]. For the audience's reference, the partitioning results of Nav-BJ and Nav-SH are visualized as shown in Fig. 7(a)-7(h), where the colored lines denote the trajectories of all road segments included in a partition. Generally, all the results indicate clear boundaries between different partitions, which are accord with to their realistic spatial localization. These results demonstrate the effectiveness of SMP for achieving spatial-aware partitioning. However, we observe that the partitioning on graphs built by CBR performs slightly better since there are fewer abrupt components in each partition, thanks to that CBR reserves more realistic traffic spatial relations among road segments when constructing graphs. Furthermore, we also illustrate the speed observations on the areas around the partition boundary as shown in Fig. 7(i). From the two examples, it is clear that in the boundary areas the road segments in different partitions report distinct speed observations. This demonstrates that SMP can identify the cut edges based on the speed observations on the traffic networks.

[4]PeMSD7 is not included in this test since this dataset does not contain related trajectory information for visualization.

Table VII presents the TF performance in terms of the two edge construction methods. It can be observed that the performance difference w.r.t. two edge construction methods is relatively small on Nav-SH compared to that on Nav-BJ. Even though the graph built by the CBR makes SMP develop better partitioning results, it cannot help improve the prediction accuracy where the performance even worsens on Nav-BJ. Furthermore, the two edge construction methods generate graphs with very different numbers of edges on Nav-SH (See Table VII), and the slight performance difference implies that our proposed approach is robust to this variation.

D. Selection of Network-partitioning Parameters

1) Sensitivity to the Number of Partitions: We first investigate the sensitivity of model performance to the number of partitions by comparing the prediction accuracy under a group setting of $S \in \{1, 2, 4, 8, 16, 32\}$, where $S = 8$ is the previously default setting and $S = 1$ represents the original STGCN approach (i.e., without network partitioning). Particularly, We perform this test using the SMP-Metis-STGCN model. From the results shown in Fig. 8(a), the best parameter of the number of partitions varies from dataset. It is $S \in \{8, 4, 4\}$ for Nav-BJ, Nav-SH and PeMSD7, respectively. It can be concluded that fine-tuning this parameter can further improve the model performance, and an overlarge one may slightly degenerate the model performance.

2) Selection of the SV Calculating Factors: We now investigate the model sensitivity to the parameters λ_{ih}, λ_{mp}, λ_{ep}, and b of the SMP approach. Specifically, we test the prediction accuracy with the base factor $b \in \{1.5, 2, 4, 6, 8, 10\}$, where $b = 2$ is the default setting. Moreover, to seek the time period whose speed observations can most significantly contribute to the final prediction performance, we set λ_{mp}, λ_{ep}, λ_{ih} to 0.8 by turns while the other two weights are set to 0.1. We use PeMSD7 to demonstrate their sensitivities due to the small size of this dataset. Offline experiments show that Nav-BJ and Nav-SH indicate a highly similar pattern. The simulation result in Fig. 8(b) indicates that the model performance is not sensitive to the exact choice of the base factor b. As regards λ, the worst accuracy is obtained when $\lambda_{ih} = 0.8$ for any b. We can conclude that λ_{mp} and λ_{ep} are preferable when calculating the SVs. This can be explained that the speed observed during idle hours usually approximates the maximum speed of a road segment. However, the maximum speeds are generally set the same for the majority of road segments in a city (e.g., 60km/h), making the calculated SVs undiversified, which further shakes the ground of the proposed Speed-Matching mechanism give full play in the network partitioning approach.

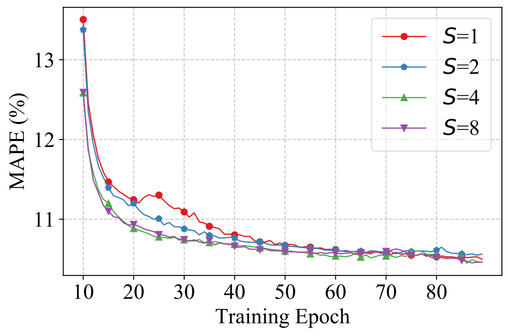
TABLE VIII
TRAINING TIME CONSUMPTION OF SMP-METIS-STGCN

Training Time (s) & Number of Included Nodes (Nav-BJ)

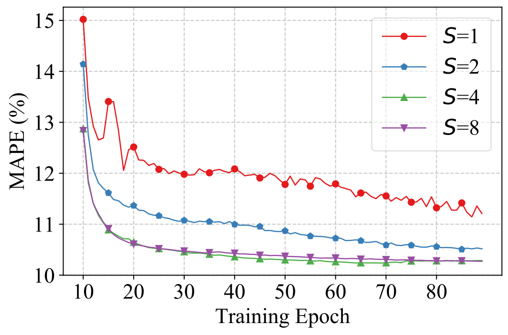
	Partition 1	Partition 2	Partition 3	Partition 4	Partition 5	Partition 6	Partition 7	Partition 8	Actual
$S = 1$	1125.1 (1159)	–	–	–	–	–	–	–	1125.1
$S = 2$	558.6 (596)	534.0 (563)	–	–	–	–	–	–	558.6
$S = 4$	297.0 (291)	294.0 (287)	289.0 (283)	300.4 (298)	–	–	–	–	300.4
$S = 8$	180.4 (149)	173.1 (142)	180.3 (149)	174.2 (142)	174.8 (143)	175.6 (144)	176.7 (145)	176.8 (145)	180.4

Training Time (s) & Number of Included Nodes (Nav-SH)

	Partition 1	Partition 2	Partition 3	Partition 4	Partition 5	Partition 6	Partition 7	Partition 8	Actual
$S = 1$	1303.4 (1335)	–	–	–	–	–	–	–	1303.4
$S = 2$	627.8 (664)	640.3 (671)	–	–	–	–	–	–	640.3
$S = 4$	325.4 (327)	325.4 (327)	337.9 (341)	335.7 (340)	–	–	–	–	337.9
$S = 8$	187.0 (162)	189.8 (164)	192.4 (168)	196.0 (171)	195.6 (171)	188.0 (162)	194.2 (170)	191.2 (167)	196.0



(a) Nav-BJ.



(b) Nav-SH.

Fig. 9. Validation MAPE versus Training epoch

E. Training Efficiency

To see the benefits of the proposed data-parallel training framework, we summarize the comparison of the learning curve and training time between the approach adopting data-parallel training (SMP-Metis-STGCN) and the approach without (original STGCN). To demonstrate a more distinguishing result, we conduct this training efficiency test on two relatively large-scale datasets, i.e., Nav-BJ and Nav-SH. Additionally, the performance comparison between $S \in \{1, 2, 4, 8\}$ is demonstrated, where the corresponding number of GPUs are used in each test. Fig. 9 depicts the learning curves on both datasets. From the results, we can observe that faster and smoother convergence can be achieved when data-parallel training is adopted (i.e., $S \in \{2, 4, 8\}$). However, the efficiency improvement is minuscule between $S = 4$ and $S = 8$, indicating a bottleneck of GPU training acceleration.

Besides, we compare the training time and present the results on Nav-BJ and Nav-SH in Table VIII, respectively. In particular, we calculate the training time on each partition (GPU), and the most extended time consumption among all the partitions is recognized as the practical time consumption under data parallelism. To investigate the correlation between the training time and the size of each partition, we also present the number of included nodes in each partition for reference. From the results, it can be observed that increasing S can contribute to the decrease in training time. SMP reduces the time consumption of each partition since the average number of nodes in each partition usually decreases with the number of partitions. Furthermore, we can find that training time has a positive correlation with the size of partitions, i.e., the larger the partition, the longer the training time. To conclude, such a data-parallel training strategy can speed up the convergences and reduce the training time to a great extent. These features are particularly essential for developing complex neural networks that may need exponentially increasing training time on large-scale traffic networks. Furthermore, in the industrial sense, crowdsourcing TF schemes in IIoT can optimize the resource allocation per these features to increase the overall efficiency, which will be investigated in future work.

VI. CONCLUSION

In this paper, based on the network-partitioning technique, we devise a domain-decomposition solution to the performance degeneration problem of GCN-based predictors on large-scale traffic forecasting. We propose a novel network-partitioning approach, namely, Speed-Matching-Partitioning (SMP). This approach utilizes both the topological feature and the traffic speed observations of traffic networks to preserve critical edges during the partition, making traffic predictors fully capture the spatial dependency in transportation networks. Besides, we introduce a data-parallel training framework to accelerate the training procedure of the partitions developed by SMP. We carry out a series of case studies based on three real-world datasets with the state-of-the-art GCN-based predictor. A significant improvement in the prediction accuracy and learning efficiency has been achieved with the proposed approach compared with other baselines. We also find that the graph representation does not have a notable influence on the model performance in practice, and all parameters employed by the proposed network-partitioning approach contribute to the TF task with different importance.

Future work can be divided into two categories. First, we plan to optimize the current graph partitioning algorithm to further improve its capacity. Second, we will investigate the interpretability of such traffic data-driven network-partitioning approaches with in-depth theoretical analysis.

References

[1] C. Lin, G. Han, J. Du, T. Xu, L. Shu, and Z. Lv, "Spatiotemporal congestion-aware path planning toward intelligent transportation systems in software-defined smart city iot," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8012–8024, 2020.

[2] A. Lei, H. Cruickshank, Y. Cao, P. Asuquo, C. P. A. Ogah, and Z. Sun, "Blockchain-based dynamic key management for heterogeneous intelligent transportation systems," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1832–1843, 2017.

[3] A. Ferdowsi, A. Eldosouky, and W. Saad, "Interdependence-aware game-theoretic framework for secure intelligent transportation systems," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[4] L. Qi, M. Zhou, and W. Luan, "A two-level traffic light control strategy for preventing incident-based urban traffic congestion," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 13–24, 2018.

[5] L. Zhu, F. R. Yu, Y. Wang, B. Ning, and T. Tang, "Big data analytics in intelligent transportation systems: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 1, pp. 383–398, 2018.

[6] F. Zhou, Q. Yang, K. Zhang, G. Trajcevski, T. Zhong, and A. Khokhar, "Reinforced spatiotemporal attentive graph neural networks for traffic forecasting," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6414–6428, 2020.

[7] 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.

[8] W. Shao, F. D. Salim, T. Gu, N.-T. Dinh, and J. Chan, "Traveling officer problem: Managing car parking violations efficiently using sensor data," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 802–810, 2017.

[9] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[10] Q. Zhang, Q. Jin, J. Chang, S. Xiang, and C. Pan, "Kernel-weighted graph convolutional network: A deep learning approach for traffic forecasting," in *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE, 2018, pp. 1018–1023.

[11] Y. Wang, D. Zhang, Y. Liu, B. Dai, and L. H. Lee, "Enhancing transportation systems via deep learning: a survey," *Transportation research part C: emerging technologies*, vol. 99, pp. 144–163, 2019.

[12] M. T. Asif, J. Dauwels, C. Y. Goh, A. Oran, E. Fathi, M. Xu, M. M. Dhanya, N. Mitrovic, and P. Jaillet, "Spatiotemporal patterns in large-scale traffic speed prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 2, pp. 794–804, 2013.

[13] 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*, 2018, pp. 3634–3640.

[14] K. Guo, Y. Hu, Z. Qian, Y. Sun, J. Gao, and B. Yin, "Dynamic graph convolution network for traffic forecasting based on latent network of laplace matrix estimation," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[15] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," in *International Joint Conference on Artificial Intelligence 2019*. International Joint Conferences on Artificial Intelligence, 2019, pp. 1907–1913.

[16] B. P. L. Lau, S. H. Marakkalage, Y. Zhou, N. U. Hassan, C. Yuen, M. Zhang, and U.-X. Tan, "A survey of data fusion in smart city applications," *Information Fusion*, vol. 52, pp. 357–374, 2019.

[17] C. N. Yahia, V. Pandey, and S. D. Boyles, "Network partitioning algorithms for solving the traffic assignment problem using a decomposition approach," *Transportation Research Record*, vol. 2672, no. 48, pp. 116–126, 2018.

[18] T. Mallick, P. Balaprakash, E. Rask, and J. Macfarlane, "Graph-partitioning-based diffusion convolutional recurrent neural network for large-scale traffic forecasting," *Transportation Research Record*, vol. 2674, no. 9, pp. 473–488, 2020.

[19] P. Johnson, D. Nguyen, and M. Ng, "Large-scale network partitioning for decentralized traffic management and other transportation applications," *Journal of Intelligent Transportation Systems*, vol. 20, no. 5, pp. 461–473, 2016.

[20] K. An, Y.-C. Chiu, X. Hu, and X. Chen, "A network partitioning algorithmic approach for macroscopic fundamental diagram-based hierarchical traffic network management," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 4, pp. 1130–1139, 2017.

[21] R. Saedi, M. Saeedmanesh, A. Zockaie, M. Saberi, N. Geroliminis, and H. S. Mahmassani, "Estimating network travel time reliability with network partitioning," *Transportation Research Part C: Emerging Technologies*, vol. 112, pp. 46–61, 2020.

[22] L. Ambühl, A. Loder, N. Zheng, K. W. Axhausen, and M. Menendez, "Approximative network partitioning for mfds from stationary sensor data," *Transportation Research Record*, vol. 2673, no. 6, pp. 94–103, 2019.

[23] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, "Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction," *Sensors*, vol. 17, no. 4, p. 818, 2017.

[24] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[25] C. Zhang, J. J. Yu, and Y. Liu, "Spatial-temporal graph attention networks: A deep learning approach for traffic forecasting," *IEEE Access*, vol. 7, pp. 166 246–166 256, 2019.

[26] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[27] 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.

[28] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, and H. Li, "T-gcn: A temporal graph convolutional network for traffic prediction," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–11, 2019.

[29] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 922–929.

[30] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *Advances in neural information processing systems*, vol. 29, pp. 3844–3852, 2016.

[31] X. Song, Y. Guo, N. Li, and L. Zhang, "Online traffic flow prediction for edge computing-enhanced autonomous and connected vehicles," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 3, pp. 2101–2111, 2021.

[32] B. Mao, F. Tang, Z. M. Fadlullah, and N. Kato, "An intelligent route computation approach based on real-time deep learning strategy for software defined communication systems," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 3, pp. 1554–1565, 2019.

[33] B. Mao, F. Tang, Z. M. Fadlullah, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "A novel non-supervised deep-learning-based network traffic control method for software defined wireless networks," *IEEE Wireless Communications*, vol. 25, no. 4, pp. 74–81, 2018.

[34] B. Mao, F. Tang, Y. Kawamoto, and N. Kato, "Ai models for green communications towards 6g," *IEEE Communications Surveys & Tutorials*, 2021.

[35] J. James, "Graph construction for traffic prediction: A data-driven approach," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[36] T. Liu, A. Jiang, X. Miao, Y. Tang, Y. Zhu, and H. K. Kwan, "Graph-based dynamic modeling and traffic prediction of urban road network," *IEEE Sensors Journal*, vol. 21, no. 24, pp. 28 118–28 130, 2021.

[37] P.-O. Fjällström, *Algorithms for graph partitioning: A survey*. Linköping University Electronic Press Linköping, 1998, vol. 3.

[38] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *The Bell system technical journal*, vol. 49, no. 2, pp. 291–307, 1970.

[39] C. M. Fiduccia and R. M. Mattheyses, "A linear-time heuristic for improving network partitions," in *19th Design Automation Conference*. IEEE, 1982, pp. 175–181.

[40] H. Ushijima-Mwesigwa, C. F. Negre, and S. M. Mniszewski, "Graph partitioning using quantum annealing on the d-wave system," in *Proceedings of the Second International Workshop on Post Moores Era Supercomputing*, 2017, pp. 22–29.

[41] Y.-H. Kim, Y. Yoon, and Z. W. Geem, "A comparison study of harmony search and genetic algorithm for the max-cut problem," *Swarm and evolutionary computation*, vol. 44, pp. 130–135, 2019.

[42] T. N. Bui and C. Jones, "A heuristic for reducing fill-in in sparse matrix factorization," in *Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing*. SIAM, 1993, pp. 445–452.

[43] B. Hendrickson and R. W. Leland, "A multi-level algorithm for partitioning graphs." *SC*, vol. 95, no. 28, pp. 1–14, 1995.

[44] S. Benhamed and S. Nait-Bahloul, "Optimization of rdf data preprocessing for metis partitioning," in *Europe and MENA Cooperation Advances in Information and Communication Technologies*. Springer, 2017, pp. 245–253.

[45] G. Kara and C. Özturan, "Algorithm 1002: Graph coloring based parallel push-relabel algorithm for the maximum flow problem," *ACM Transactions on Mathematical Software (TOMS)*, vol. 45, no. 4, pp. 1–28, 2019.

[46] D. Delling, A. V. Goldberg, I. P. Razenshteyn, and R. F. F. Werneck, "Graph partitioning with natural cuts," in *25th IEEE International Symposium on Parallel and Distributed Processing*. IEEE, 2011, pp. 1135–1146.

[47] H. Yuan and G. Li, "Distributed in-memory trajectory similarity search and join on road network," in *2019 IEEE 35th international conference on data engineering (ICDE)*. IEEE, 2019, pp. 1262–1273.

[48] K. Guo, Y. Hu, Z. Qian, H. Liu, K. Zhang, Y. Sun, J. Gao, and B. Yin, "Optimized graph convolution recurrent neural network for traffic prediction," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–12, 2020.

[49] H. Yu, Z. Wu, S. Wang, Y. Wang, and X. Ma, "Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks," *Sensors*, vol. 17, no. 7, p. 1501, Jun. 2017.

[50] G. Li, M. Mueller, G. Qian, I. C. Delgadillo Perez, A. Abualshour, A. K. Thabet, and B. Ghanem, "Deepgcns: Making gcns go as deep as cnns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021.

[51] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh, "Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 257–266.

[52] A. Furno, N.-E. El Faouzi, R. Sharma, and E. Zimeo, "Two-level clustering fast betweenness centrality computation for requirement-driven approximation," in *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 2017, pp. 1289–1294.

[53] R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman, and D. Cohen-Or, "Meshcnn: a network with an edge," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 4, pp. 1–12, 2019.

[54] A. Kirkley, H. Barbosa, M. Barthelemy, and G. Ghoshal, "From the betweenness centrality in street networks to structural invariants in random planar graphs," *Nature communications*, vol. 9, no. 1, pp. 1–12, 2018.

[55] U. Brandes and C. Pich, "Centrality estimation in large networks," *International Journal of Bifurcation and Chaos*, vol. 17, no. 07, pp. 2303–2318, 2007.

[56] G. Karypis, "Metis: Unstructured graph partitioning and sparse matrix ordering system," *Technical report*, 1997.

[57] J. L. G. García, R. Yahyapour, and A. Tchernykh, "Graph partitioning for fem applications: Reducing the communication volume with dshem," in *2019 International Conference on High Performance Computing & Simulation (HPCS)*. IEEE, 2019, pp. 779–786.

[58] J. Hammersley, *Monte carlo methods*. Springer Science & Business Media, 2013.

[59] J. J. Q. Yu and J. Gu, "Real-time traffic speed estimation with graph convolutional generative autoencoder," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 10, pp. 3940–3951, 2019.