Graph Neural Network-based Clustering Enhancement in VANET for Cooperative Driving

Hang Hu, Myung J. Lee
Department of Electrical Engineering
City College, City University of New York
New York, NY, USA, 10031
Emails: hhu002@citymail.cuny.edu, mlee@ccny.cuny.edu

***Abstract*—The significantly increasing number of vehicles brings convenience to daily life while also introducing significant challenges to the transportation network and air pollution. It has been proved that platooning/clustering-based driving can significantly reduce road congestion and exhaust emissions and improve road capacity and energy efficiency. This paper aims to improve the stability of vehicle clustering to enhance the lifetime of cooperative driving. Specifically, we use a Graph Neural Network (GNN) model to learn effective node representations, which can help aggregate vehicles with similar patterns into stable clusters. To the best of our knowledge, this is the first generalized learnable GNN-based model for vehicular ad hoc network clustering. In addition, our centralized approach makes full use of the ubiquitous presence of the base stations and edge clouds. It is noted that a base station has a vantage view of the vehicle distribution within the coverage area as compared to distributed clustering approaches. Specifically, eNodeB-assisted clustering can greatly reduce the control message overhead during the cluster formation and offload to eNodeB the complex computations required for machine learning algorithms. We evaluated the performance of the proposed clustering algorithms on the open-source highD dataset. The experiment results demonstrate that the average cluster lifetime and cluster efficiency of our GNN-based clustering algorithm outperforms state-of-the-art baselines.**

***Index Terms*—vehicular ad hoc network (VANET), graph neural networks (GNNs), clustering algorithm, stability, cooperative driving**

I. Introduction

With the rapid development of the Automobile Industry and Urbanization, there are more and more vehicles on the roads. It is well-established that more than one billion vehicles have been registered globally, expected to grow in the following decades. Consequently, the problems associated with the increased number of vehicles have become more severe, including traffic congestion, traffic accidents, energy waste, and air pollution. In the United States, traffic congestion costs drivers more than $100 billion annually due to wasted fuel and lost time [1]. In addition, exhaust emissions caused by traffic congestion are considered a key contributor to air pollution and a major haze component in many cities. For instance, the most significant source of greenhouse gases in the USA comes from the transportation sector, which accounts for 29% of total greenhouse gas emissions [2].

While the construction of roads can increase traffic capacity and reduce traffic congestion to some extent, it is unsustainable due to the enormous construction costs and limited land availability, especially in urban areas. An effective way to solve these problems is to change the driving pattern from individual driving to platoon driving [3] [4]. In general, a platoon-based driving pattern is a cooperative driving pattern of a group of vehicles with common interests, where one vehicle follows another and keeps a small and almost constant distance from the preceding vehicle to form a platoon.

The cooperative platoon-based driving pattern can significantly enhance road capacity, safety, energy efficiency, and collaborative environment. However, establishing and maintaining stable clusters or platoons in Connected Vehicles Networks are challenging because of the heterogeneous and drastically changing traffic scenarios. Moreover, in order to maintain multicast group communication in cooperative platoon-based driving among cluster members (CMs), the stable clustering algorithm is crucial. As all future vehicles can access base stations (BS or eNodeB), more efficient clustering is possible but yet to be prevalent with the help of cellular infrastructure, i.e., BS.

Most vehicular clustering algorithms have taken a distributed approach based on Dedicated Short-Range Communications (DSRC) without infrastructure support. Thus, they rely mainly on periodic HELLO messages exchanged among vehicles. Moreover, the most vital part of the clustering algorithm is Cluster Head (CH) selection [5] [6] [7], which is essential for the stability of the cluster lifetime and the control message overhead involved in forming and maintaining these clusters. Weight-based algorithms are widely used for CH selection [8] [9]. Each vehicle calculates a metric according to messages received from its neighbors. The metric represents the fitness to serve as a CH and broadcast to each vehicle's neighbors. The vehicle with the highest metric weight will act as a CH among nearby vehicles. The metric can generally be related to network metrics, such as degree of connectivity, link stability, and density, and mobility metrics, such as position, velocity, acceleration, and destination.

This distributed clustering strategy tends to increase the control message overhead compared with centralized strategy. Additionally, the weight-based algorithm of CH selection needs to manually adjust the combination of hyper-parameters among multiple metrics. Last but not least, existing vehicular clustering algorithms are not intelligent and learnable to adapt to different traffic scenarios. As a result, they are not easy to satisfy the requirement of the evolving Intelligent Transportation System (ITS).

In this paper, we propose to use Graph Neural Network (GNN) [10] [11], which fits naturally to solve clustering type of graph problem. To the best of our knowledge, applying GNN to solve the clustering problem in Vehicular Ad hoc Network (VANET) is the very first attempt. GNN uses both feature and graph information and usually achieves better performance than methods leveraging single feature or graph structure such as k-means [17] or Spectral Clustering [12]. Our proposed algorithm is a centralized approach and offloads the computation of GNN to BS instead of executing it at individual vehicles, alleviating the computational burden from vehicular nodes. We note that a base station has a vantage view

of the vehicle distribution within the coverage area compared to distributed approaches. Specifically, eNodeB-assisted clustering can greatly reduce the control message overhead during cluster formation. In practice, a trained GNN model located at BS or edge cloud utilizes the collected vehicle information to perform clustering and informs CHs to formulate cooperative driving patterns to improve traffic efficiency. Fig. 1 illustrates the framework of vehicular network clustering.

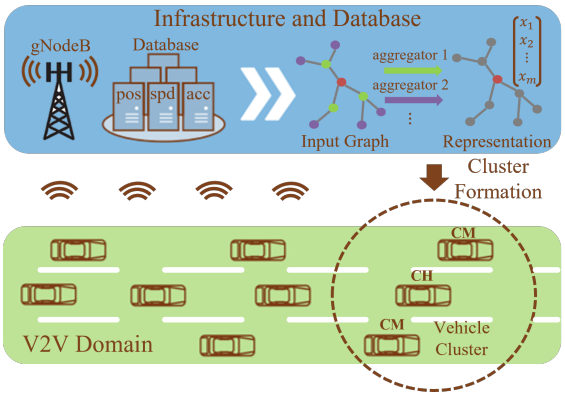

Fig. 1. Architecture of Vehicular Network Clustering.

II. Related Works

Clustering is introduced to improve routing scalability and reliability and enhance the stability of the collaborative environment by exploiting the formation of hierarchical network structures. By grouping vehicles together in the consideration of correlated spatial distribution and relative velocity, these cluster groups can serve as the foundation for accident or congestion detection, information dissemination, and entertainment in the applications of ITS.

A. Distributed Clustering Approaches

The earliest VANET clustering methods are derived from Mobile Ad hoc Network (MANET) clustering approaches to facilitate the distribution of network resources. A majority of them have taken a distributed approach based on DSRC without infrastructure support. Many existing VANET clustering algorithms focus on optimally selecting CHs because the stability of a cluster depends primarily on the selection of the CH. Weight-based metrics are the key to the most clustering strategies: position [13], speed [14], destination [15], and multiple metrics [16]. Most conventional distributed approaches still incur high communication overhead and prove inefficient in a highly dense and dynamic environment.

B. Machine Learning based Clustering Approaches

Clustering algorithms began adopting machine learning to overcome the required complex computation in distributed clustering. K-means algorithm [17] is the most frequently used machine learning algorithm in VANET. Some k-means variant algorithms [18] are proposed to enhance initial centroid selection to boost performance. Often, the fuzzy logic inference is integrated with a machine learning algorithm to enhance the stability of a cluster [19] by predicting the future speed and the positions of CMs. Nevertheless, the design of fuzzy rules needs much domain knowledge, and fuzzy logic does not have the learning ability as is well-known. Some researchers recently proposed using Spectral Clustering, which is related to Eigenvalue Decomposition (EVD) on the normalized graph Laplacian matrix, to enhance clustering stability in VANET [20].

C. GNN based Clustering Approaches

Recently, research on analyzing graphs with machine learning has been receiving more attention because of the great expressive power of graph data, which contains rich relation information among elements. Hence, GNNs have been proposed to solve the non-Euclidean domain problem. In GNNs, node clustering divides the nodes into several disjoint groups where similar nodes should be in the same group. [21] has applied graph autoencoder (GAE) to node clustering (citation network) by an unsupervised learning framework. Even though GNNs have many applications across different tasks and domains, applying GNN to solve the clustering problem in VANET is the very first attempt.

In this paper, our goal is to enhance the vehicle system's stability and optimize the average lifetime of all clusters. The problem of clustering is innovatively transformed into aggregating vehicles with similar node representations (embeddings) in the same cluster as learned by the GNN model. Specifically, a partitioning method optimally divides the vehicle nodes into groups with a minimum intra-cluster dissimilarity. Our proposed algorithm coincides with the goal of VANET clustering, which is to encourage vehicles with similar motion patterns, such as similar position, velocity, acceleration, etc., to form a cluster.

III. GNN-based Clustering In VANET

In this section, we demonstrate how to develop a GNN-based clustering scheme that collects vehicle feature as its input and node representation as its output.

A. Graph Construction

The interconnections among vehicles driving on the road can be formulated as an undirected homogeneous dynamic graph. To this end, we propose to use GNN, which fits naturally to solve clustering type of graph problem and uses both feature and graph information. We use the raw vehicle feature as the node feature of the graph. A vehicle feature of vehicle node v_i at time t is $x_i(t) = \{\boldsymbol{s}_i, \boldsymbol{p}_i, \boldsymbol{a}_i, l_i, w_i\}$, where $\boldsymbol{s}_i$ is the speed, $\boldsymbol{p}_i$ is the position, $\boldsymbol{a}_i$ is the acceleration, l_i and w_i are the length and width of vehicle v_i. In the following, we use the subscript i to denote vehicle node v_i.

The vehicle interconnection metric is designed to weigh the similarity between the movement patterns of two vehicles. In our model, the vehicle interconnection metric is calculated by the improved force-directed algorithm designed based on virtual forces [22], which is inspired by Coulomb's Law to select CH and create stable clusters. The force-directed algorithm assigns the forces on the edges in the VANET graph. Note that the force also represents the weight between any two connecting vehicle nodes. The most straightforward way is to assign force as if the edges were springs and the nodes were electrically charged particles. The entire network is modeled as a physical system. The forces are applied to the vehicle nodes, pulling them closer together or pushing them further away. Every vehicle node exerts a force $\boldsymbol{F}$ on its neighbors according to their distance and relative velocities. Fig. 2 shows the neighbored vehicle nodes apply relative forces to the vehicle v_i.

A positive force between two nodes indicates that the pair of nodes are moving in the same direction. In contrast, a negative force between two nodes means that the vehicles are moving in opposite directions. In our model, the relative force is always positive since we only consider the vehicles are moving in the

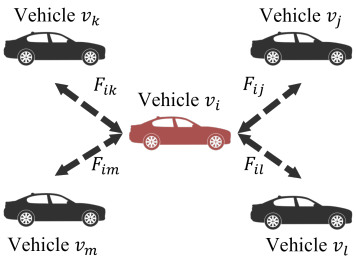

Fig. 2. Neighbored forces applied to vehicle v_i.

same direction, which facilitates the stability of VANET. The greater the positive forces among nodes are, the more similar the moving pattern is.

Here we explain how to calculate the pairwise relative force $\boldsymbol{F_{ij}}$ for every neighbor applied. Naturally, the relative force is decomposed along the x-axis and the y-axis. We can obtain relative force $\boldsymbol{F_{ij}}$ according to equations defined as:

$$F_{ijx} = k_{ijx}\frac{q_i q_j}{D_{ij}^2}; F_{ijy} = k_{ijy}\frac{q_i q_j}{D_{ij}^2} \tag{1}$$

$$D_{ijx}(t) = x_i - x_j; D_{ijx}(t+dt) = x_i + dx_i - x_j - dx_j \tag{2}$$

$$D_{ijy}(t) = y_i - y_j; D_{ijy}(t+dt) = y_i + dy_i - y_j - dy_j \tag{3}$$

$$k_{ijx} = \frac{1}{1 + |D_{ijx}(t+dt) - D_{ijx}(t)|dt} \tag{4}$$

$$k_{ijy} = \frac{1}{1 + |D_{ijy}(t+dt) - D_{ijy}(t)|dt} \tag{5}$$

$$q_i = q_j = \begin{cases} R - D_{ijx}(t), & \text{if } D_{ijx}(t) \le D_{ijx}(t+dt) \\ R + D_{ijx}(t), & \text{if } D_{ijx}(t) > D_{ijx}(t+dt) \end{cases} \tag{6}$$

$$||\boldsymbol{F}_{ij}||_2 = \sqrt{F_{ijx}^2 + F_{ijy}^2} \tag{7}$$

Where F_{ijx} and F_{ijy} are the relative forces along the x-axis and y-axis. k_{ijx} and k_{ijy} are the relative mobility parameters along the x-axis and y-axis, respectively. q_i and q_j represent relative maintenance parameters indicating that how far they are beyond communication distance. D_{ij} is the current distance among the nodes. $D_{ijx}(t)$ and $D_{ijy}(t)$ are the distance between two nodes along the x-axis and y-axis at time t. Similarly,$D_{ijx}(t+dt)$ and $D_{ijy}(t+dt)$ are the distance between two nodes along the x-axis and y-axis at time $t+dt$. x_i and y_i represent the x-axis and y-axis position of node v_i. dx_i and dy_i are the position increment in time dt on the x-axis and y-axis of node v_i. R is the transmission range. This force F_{ij}will be used as weight to propagate information in Eq. (8).

B. Design of GNN Clustering Algorithm

Having obtained a customized graph dataset in the last section, we present our GNN-based clustering algorithm here. In general, our GNN model comprises four layers, including an input layer, two SAGE (SAmple and aggreGatE) Convolutional layers (SAGEConv), and an output layer. The dimension of the input layer is the vehicle feature, and the output dimension is predefined (e.g., 4 in our experiment). The core layer of our GNN is inductive SAGE Convolutional layers.

SAGEConv layer is derived from graphSAGE [23], a general inductive framework that leverages node feature information to generate node embeddings for each node efficiently. This inductive capability can generalize to operate on evolving graphs and unseen nodes. Specifically, the SAGEConv layer generates node embeddings by aggregating information from their local neighbors. The detailed visual illustration of the SAGEConv layer is shown in Fig. 3 The aggregation of SAGEConv is formulated as:

$$h_i^k = \sigma\Big(W^k \cdot \frac{1}{|N_i|}\sum_{j\in N_i}(h_j^{k-1} \cdot F_{ij})\Big) \tag{8}$$

Where h_i^k is the embedding of node v_i in the kth layer. N_i is the neighborhood set connected to node v_i. W^k is the learnable weight parameters of fully connected layer k. F_{ij} is the weight to aggregate message. $\sigma(\cdot)$ is the activation function ReLU. The number of GNN layers, namely search depth, is also the number of neighbors of hops aggregated information by the target node. In a k-layer GNN, nodes are able to collect information from the neighbors of the k-hops.

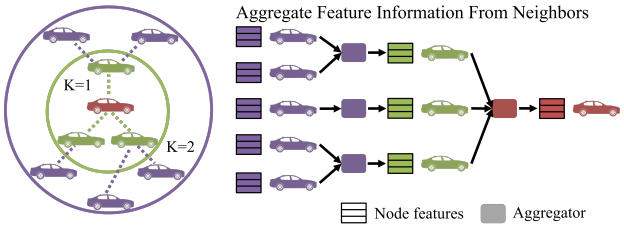

Fig. 3. Visual illustration of the SAGEConv layer.

On the other hand, early node embedding approaches are inherently transductive and directly optimize the embedding for each node using matrix-factorization-based objectives [25]. Consequently, they do not naturally generalize to unseen data since they predict nodes in a single, fixed graph. Combined with our application, the characteristic of the SAGEConv satisfies the dynamic traffic scenario. Our proposed GNN-based clustering algorithm is shown in Algorithm 1, where $J_G(z_i)$ is the objective function discussed in section III part C. Due to the deeper layers (2nd loop), this process is iterative, and the nodes gradually acquire more and more information from further away from the graph. We use 2 SAGEConv layers (i.e., search depth $K = 2$). There are several choices of aggregator architectures, such as Mean aggregator, Long Short-Term Memory (LSTM) aggregator, and Pooling aggregator. Here we choose Mean aggregator, which is a simple but with significant gain in performance to compute the embedding. The output is a low-dimensional vector embedding of the nodes. It already proves to be extremely useful for feature input for various downstream tasks such as classification, prediction, and clustering.

C. Model Training

To learn effective and useful representations in a completely unsupervised learning fashion, a graph-based loss function $J_G(z_i)$ is applied to the output representations z_i, and the weight matrices W^k is tuned via backward propagation. This loss function is defined as:

$$J_G(z_i) = -\sum_{i,j\in V}(y_{ij}\log(\hat{y}_{ij}) + (1-y_{ij})\log(1-\hat{y}_{ij})) \tag{9}$$

$$\hat{y}_{ij} = \sigma(z_j^T z_i), \tag{10}$$

where j is a node v_j which is within the transmission range to node v_i. $\hat{y}_{ij}$ denotes the probability of an edge with logits between the node i and j.

Specifically, we sample the edges in a graph as positive examples and non-existent edges (i.e., node pairs with no

edges between them) as negative examples. Positive and negative examples have the same number. Then, the positive and negative examples form positive and negative graphs, respectively. Along with the forward propagation, we can calculate node representations via the GNN model and apply them to the positive and the negative graphs for computing pairwise probability among nodes. We can calculate the loss and update model parameters via stochastic gradient descent along with the backward propagation.

Algorithm 1: GNN-based Clustering Algorithm

Input : Graph $G(V, E)$, edge weight F_{ij}
Vehicle feature $x_i, \forall i \in V$
Search depth k, $\forall k \in \{1, ..., K\}$
Weight matrices W^k
Neighborhood set N_i
Number of iterations T
Graph-based loss function J_G

Output: Clustering assignments

$h_i^0 \leftarrow x_i$;
for $t = 1$ *to* T **do**
  for $k = 1$ *to* K **do**
    for $i \in V$ **do**
      $h_i^k \leftarrow \sigma\Big(W^k \cdot \frac{1}{|N_i|} \sum_{j \in N_i} (h_j^{k-1} \cdot F_{ij})\Big)$;
      $z_i^k \leftarrow h_i^k / ||h_i^k||_2$;
    end
  end
  Calculate $J_G(z_i)$ and update via stochastic gradient descent
end
Run k-means on output embeddings z_i to obtain final clustering results

IV. Performance Evaluation

In this experimental section, we first introduce the benchmark datasets and experimental parameter settings used in the experiments. After that, we evaluate the training of the proposed GNN model to validate that our model can learn useful and effective node representations. In addition, we show the baseline algorithms used in the results. Finally, we evaluate the metric performance used for VANET clustering between our algorithm and the baseline algorithms.

We implement our simulation platform in Python framework with PyTorch [26] and Deep Graph Library (DGL), which is a Python package built for easy implementation of graph neural network model family [24]. In addition, we conduct the following experiments on a computer with a 2.21GHz Intel Core i7-8750H CPU, 16GB Memory. Our proposed scheme is used for highway scenarios. Furthermore, the traffic model and the evaluations are based on real traffic data.

A. Datasets and Parameter Settings

1) Datasets: We construct a customized graph dataset for model training on the open-source highD dataset [27]. The highD dataset is new naturalistic vehicle trajectory recordings on German highways. A camera-equipped drone recorded the traffic with a 25fps frame rate at six different locations, and it covered a road segment of about 420 m length. The locations vary by the number of lanes, speed limits, and traffic density. Using state-of-the-art computer vision algorithms for semantic segmentation, the authors have estimated every pixel of each frame, whether it belongs to a vehicle or the background. The positioning error is typically less than ten centimeters. It is convenient to obtain traffic information, including vehicle trajectory, vehicle type, size, and maneuvers. We extract vehicle feature x_i including speed $\boldsymbol{s}_i$, position $\boldsymbol{p}_i$ acceleration $\boldsymbol{a}_i$, length l_i and width w_i of vehicles and standardize features by removing the mean and scaling to unit variance.

We choose sequence 13 recording to build the graph dataset, which involves the largest number of vehicles, since machine learning algorithms generally have a distinct advantage when dealing with a large amount of data. With the graph construction algorithm in section III part A, we generated 1000 training graphs and 210 testing graphs where we only consider cars instead of trucks since cars and trucks might have different driving patterns. For simplicity, we consider the same type of cars. The graph construction of data frame 32 is shown in Fig. 4. The green triangle stands for the vehicles. The brown dashed lines represent the edges among the vehicles. The blue dashed lines indicate lanes.

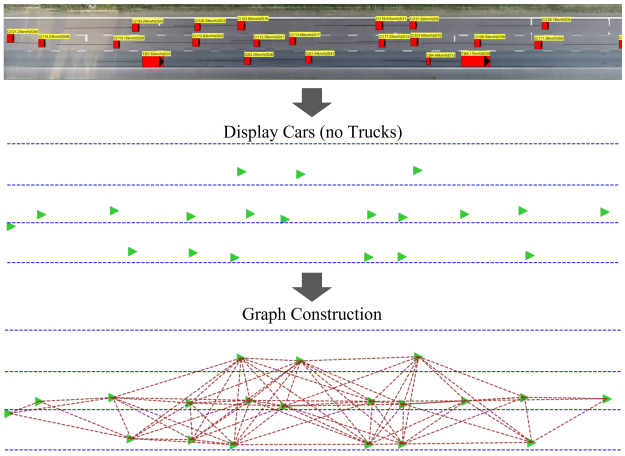

Fig. 4. Visual illustration of Graph Construction.

2) Parameter Settings: The dimension of the input layer is 8 (i.e., vehicle feature dimension). We set the dimension of the hidden layer and output layer to low-dimensional as 4. The maximum epoch for training is 400. To avoid overfitting, we apply early stopping. If the number of times that the validation loss is greater than the minimum loss exceeds a threshold (e.g., 150 in our experiment), the training will stop. We randomly sample the edges on each graph to form the training and validation sets. The edge ratio in the training set to the validation set is 9 to 1. We select ADAM with a learning rate of 0.003 as the optimization strategy. In addition, for reproducibility, we set a random seed (42069).

B. Model Training and Clustering Results

In order to evaluate the performance of node representations of our GNN model, we employ three metrics: Binary Cross Entropy Loss, Accuracy, and Area Under Curve (AUC). Binary Cross Entropy Loss is the objective function (i.e., Eq. (9)). AUC stands for the area under Receiver Operating Characteristic (ROC) curve, and the higher value indicates better performance. Accuracy is the ratio of the correctly classified elements to the total number of elements.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{11}$$

where TP, TN, FP and FN are true positive, true negative, false positive and false negative, respectively.

After 1.5 hours of computation, the training of our GNN model is completed. The results show that the training and validation losses are 0.041 and 0.084, respectively. The training and validation accuracy are 0.986 and 0.969, respectively. In addition, the loss and accuracy on the testing graphs are 0.063 and 0.978, respectively. Here validation set is used to check the model convergence during training and avoid overfitting. Moreover, the testing set is applied to evaluate model generalization capability, and the AUC on testing graphs gets a good score of 0.998. Thus, we conclude that our GNN model can learn useful and predictive node representations. The training loss and accuracy are shown in Fig. 5. We have obtained a trained GNN model that can be used to learn useful node representations. Furthermore, we apply the trained GNN model on a graph and then obtain the clustering results by using k-means on node representations of the graph.

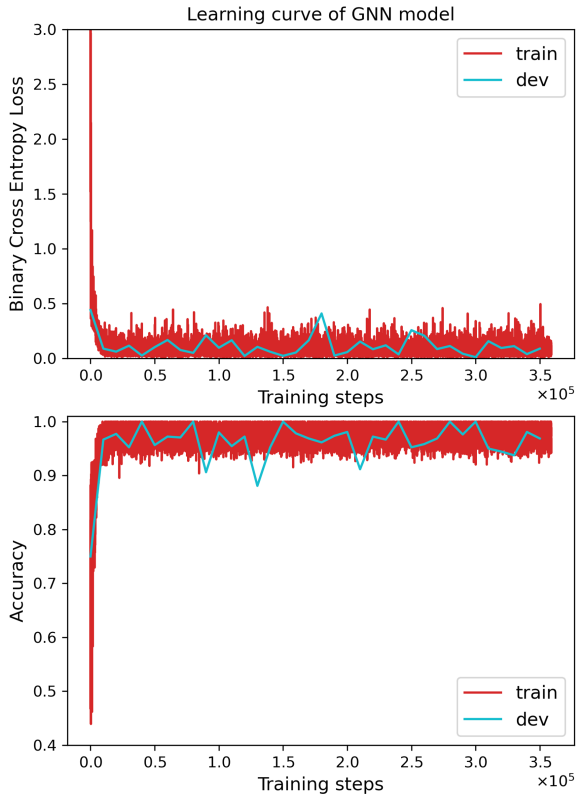

Fig. 5. Loss and accuracy curve during training.

In order to see the clustering results, we refer to [18] to select the number of clusters:

$$n = \left\lceil \frac{L}{2R} \right\rceil, \tag{12}$$

where L denotes the length of the road. R denotes the transmission radius of the vehicle. The length of the road segment L is about 420 m, and the transmission range R is defined as 100 m. Therefore, the number of clusters is $n = 3$. The clustering result on testing data frame 66 is shown in Fig. 6. The colored triangles represent the 3 clusters formed by our proposed clustering algorithm. The red dashed circles mark the CHs.

C. Baseline Algorithms

We used the following clustering methods as the baseline algorithms in our comparisons.

1) Method using features only: k-means is traditional clustering algorithms [17]. Here we run k-means on our original vehicle feature as a benchmark.

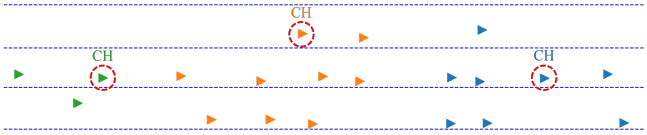

Fig. 6. GNN-based clustering results.

2) Method using graph structure only: Spectral Clustering [20] uses the adjacency matrix as the input similarity matrix to perform dimensionality reduction before clustering and is widely used in graph clustering.

3) Method using both features and graph: We also compare our proposed algorithm with the Graph Autoencoder (GAE) based clustering algorithm [21], which is an unsupervised learning network embeddings by encoding nodes/graphs into a latent vector space and reconstructing graph data from the encoded information. Since GAE is a matrix-factorization-based method, it can only be used for fixed graphs. Thus, we trained every graph before evaluating them.

D. VANET Performance Evaluation and Results

We evaluate the clustering performance in VANET based on the highD dataset by our GNN-based algorithm and baseline algorithms. Since the highD dataset covers a road segment, we can only obtain a limited period of tracking time and distance. The coverage of the road segment is about 420 m length. Each vehicle is visible for a median duration of 13.6 s.

We run our algorithm on 210 testing graphs. Then, we take each testing graph as the initial frame and track each initial frame. We read the data frames backward until all vehicles on the initial frame disappear in the road segment. In this finite process, we record the number of vehicles out of the transmission range of corresponding CHs and leaving the initial clusters. In the whole process, we tracked 67,119 frames, and it involved 4558 vehicles. The number of vehicles breaking the initial clusters is shown in Fig. 7. In the above and following process, we both run ten times to eliminate randomness and then calculate the mean and standard deviation. The results show that our algorithm corresponds to the minimum number of vehicles breaking the initial clusters.

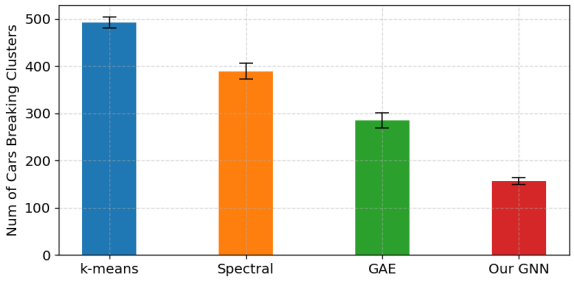

Fig. 7. Number of vehicles breaking the initial clusters.

On this basis, we evaluate the average cluster lifetime on the testing graphs, which indicates the time span that the CMs keep unchanged. If a cluster whose CMs remain unchanged during our whole trace on the highD dataset, its average cluster lifetime is this whole trace time. As shown in Fig. 8, the average cluster lifetimes are 11.039±0.038 s, 11.231±0.099 s, 11.837±0.110 s, 12.069±0.037 s with confidence 95% for k-means, Spectral Clustering, GAE, and our GNN. Clearly, the longer the average cluster lifetime is, the more stable the cluster is. Compared with baseline algorithms, our GNN-based clustering algorithm has the longest average cluster

lifetime, reaching 12 s, which is very close to the median duration of 13.6 s. Furthermore, methods using both features and graph structure, i.e., GAE and our GNN, are more stable than methods using either features or graph structure only.

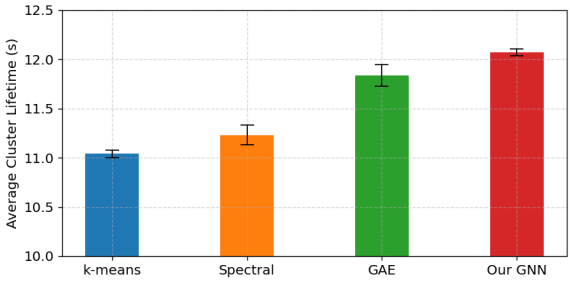

Fig. 8. Average cluster lifetime.

We also performed a quantitative analysis of coverage percentage (CP). In this paper, the CP is defined as:

$$CP = \frac{N - N_{Iso}}{N}, \tag{13}$$

where N_{Iso} is the number of isolated vehicles which do not belong to any cluster. As shown in Fig. 9, the coverage percentage of four algorithms are 86.062±0.455%, 98.383±0.173%, 97.734±0.517%, 98.927±0.111% with confidence 95%, respectively. The results indicate our GNN-based algorithm's cluster efficiency outperforms baselines.

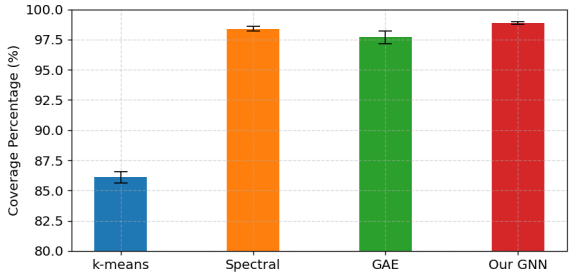

Fig. 9. Coverage percentage.

V. Conclusion

In this paper, our research aimed to establish and maintain stable clusters for cooperative driving in VANET. Based on quantitative and qualitative analysis on the open-source highD traffic dataset, it can be concluded that our GNN-based clustering algorithm using both features and graph structure outperforms the baseline algorithms. Moreover, this is the very first attempt to solve the VANET clustering problem by applying GNN. Our intelligent and learnable GNN model gives the possibility to adapt to different traffic scenarios.

As future works, we plan to study other traffic scenarios like urban environment and Simulation of Urban MObility (SUMO) for long-term performance since the highD dataset is limited in total tracking time.

Acknowledgment

This research is supported by NSF IRNC Grant No. 2029295.

References

[1] Transport Topics. "Traffic congestion costs billions in wasted fuel, time, report says,"Mar. 28, 2014. [Online]. Available: http://www.ttnews.com/articles/basetemplate.aspx?storyid=29007.

[2] EPA (United States Environmental Protection Agency). Sources of Green-house Gas Emissions. Greenh. Gas Emiss. Accessed 3 Mar 2020. https://www.epa.gov/ghgemissions/sources-greenhouse-gas-emissions.

[3] P. Kavathekar and Y. Chen, "Vehicle platooning: A brief survey and categorization," in Proc. 7th ASME/IEEE Int. Conf. MESA/ASME DETC/CIE, 2011, pp. 1–17.

[4] D. Jia, K. Lu, J. Wang, X. Zhang, and X. Shen, "A Survey on Platoon-Based Vehicular Cyber-Physical Systems," IEEE Communications Surveys & Tutorials, 18(1), 2016, pp.263-284.

[5] C. Cooper, D. Franklin, M. Ros, F. Safaei, and M. Abolhasan, "A comparative survey of VANET clustering techniques," IEEE Communications Surveys & Tutorials, 19(1), 2016, 657-681.

[6] A. Katiyar, D. Singh, and R. S. Yadav, "State-of-the-art approach to clustering protocols in vanet: A survey," Wireless Networks, 26(7), 2020, 5307-5336.

[7] M. Ren, J. Zhang, L. Khoukhi, H. Labiod, and V. Vèque, "A review of clustering algorithms in VANETs," Annals of Telecommunications, 1-23, 2021.

[8] A. Daeinabi, A. G. P. Rahba, and A. Khademzadeh, "VWCA: An efficient clustering algorithm in vehicular ad hoc networks," J. Netw. Comput. Appl., vol. 34, no. 1, 2011, pp. 207-222.

[9] R. Chai, B. Yang, L. Li, X. Sun and Q. Chen, "Clustering-based data transmission algorithms for VANET," International Conference on Wireless Communications and Signal Processing, Hangzhou, 2013, pp. 1-6.

[10] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," IEEE transactions on neural networks and learning systems, 32(1), 2020, 4-24.

[11] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, M. Sun, and et al., "Graph neural networks: A review of methods and applications," AI Open, 1, 2020, 57-81.

[12] U. Von Luxburg, "A tutorial on spectral clustering," Statistics and computing, 17(4), 2007, 395-416.

[13] X. Bao, H. Li, G. Zhao, L. Chang, J. Zhou, and Y. Li, "Efficient clustering V2V routing based on PSO in VANETs," Measurement, 152, 2020, 107306.

[14] M. Ren, L. Khoukhi, H. Labiod, J. Zhang, and V. Veque, "A mobility- based scheme for dynamic clustering in vehicular ad-hoc networks(VANETs)," Vehicular Communications, 9, 2017, 233-241. Communications and Information Technologies (ISCIT), 2014, pp.233-237

[15] A. Bello Tambawal, R. Md Noor, R. Salleh, C. Chembe, and M. Oche, "Enhanced weight-based clustering algorithm to provide reliable delivery for VANET safety applications," PloS one, 14(4), 2019, e0214664.

[16] B. Azat, B and T. Hong, "Destination based stable clustering algorithm and routing for vanet," Journal of Computer and Communications, 8(01), 2020, 28.

[17] N. Taherkhani and S. Pierre, "Centralized and localized data congestion control strategy for vehicular ad hoc networks using a machine learning clustering algorithm. IEEE Transactions on Intelligent Transportation Systems,"17(11), 2016, 3275-3285.

[18] R. Chai, X. Ge, and Q. Chen, "Adaptive K-harmonic means clustering algorithm for VANETs," International Symposium on computing, networking and communications (WiMob), 2012, pp. 593-599.

[19] M. A. Saleem, S. Zhou, A. Sharif, T. Saba, M. A. Zia, A. Javed, M. Mittal, and et al., "Expansion of cluster head stability using fuzzy in cognitive radio CR-VANET," IEEE Access, 7, 2019, 173185-173195.

[20] G. Liu, N. Qi, J. Chen, C. Dong, and Z. Huang, "Enhancing clustering stability in VANET: A spectral clustering based approach. China Communications," 17(4), 2020, pp. 140-151.

[21] T. N. Kipf, and M. Welling, "Variational graph auto-encoders," arXiv preprint arXiv:1611.07308, 2016.

[22] L. A. Maglaras, and D. Katsaros, "Distributed clustering in vehicular networks," IEEE 8th international conference on wireless and mobile computing, networking and communications (WiMob), 2012, pp. 593-599.

[23] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," In Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017, pp. 1025-1035.

[24] M. Wang, L. Yu, D. Zheng, Q. Gan, Y. Gai, Z. Ye, Z. Zhang, and et al., "Deep Graph Library: Towards Efficient and Scalable Deep Learning on Graphs," arXiv preprint arXiv:1909.01315, 2019.

[25] S. Cao, W. Lu, and Q. Xu. Grarep: Learning graph representations with global structural information. In KDD, 2015.

[26] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, S. Chintala, and et al., "Pytorch: An imperative style, high-performance deep learning library," Advances in neural information processing systems, 32, 2019, 8026-8037.

[27] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein, "The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems," International Conference on Intelligent Transportation Systems (ITSC), 2018, pp. 2118-2125.