Clustering-learning-based Long-term Predictive Localization in 5G-Envisioned Internet of Connected Vehicles

Kai Lin*, *Senior Member, IEEE*, Yihui Li, Jing Deng, *Fellow, IEEE*, Pasquale Pace, *Member, IEEE*, and Giancarlo Fortino, *Senior Member, IEEE*

***Abstract*—Localization services play an important role in Internet of Connected Vehicles (IoCV) and vehicle predictive localization information can greatly improve traffic efficiency and reduce accidents. However, a huge amount of computing and communication overhead is required to obtain such information by traditional methods. In this work, we propose a Behavior-based Clustering Method (BCM) to analyze the behavioral correlation between vehicles and classify them into different clusters. Based on BCM results coupled with a deep learning model, we further propose a Clustering-learning-based Long-term Predictive Localization (CLPL) algorithm to predict vehicles' future location distribution. In the proposed CLPL algorithm, all the traffic roads are divided into consecutive small segments in order to pinpoint vehicles' precise current locations and to obtain long-term predictions. Extensive simulations, notably involving real dataset, have been carried out to evaluate BCM and CLPL in terms of several performance criteria including matching rates. The analysis of the results validated how the designed methods can predict vehicle location much more accurately than existing algorithms.**

***Index Terms*—Internet of Connected Vehicles; Clustering learning; Behavior analysis; Vehicle localization.**

I. Introduction

With the rapid development of mobile computing and communication technologies, the integration of Internet of Connected Vehicles (IoCV) and 5G wireless network promises many driving-enhancing services for drivers and passengers [1]. Vehicles and road side units (RSUs) are the basic components of IoCV, and they are able to communicate with each other and access 5G wireless network services. As an application of Internet of Things (IoT) in intelligent transportation systems, IoCV is mainly used in urban traffic environments to support network access for drivers, passengers, and traffic managements [2], [3].

The problem of vehicle predictive localization has received significant attentions in the last few years. For instance, Jo *et al.* [4] proposed a unified vehicle tracking and behavior reasoning algorithm, which applies a roadway geometry constraint to improve the dynamic state evaluation and the behavior classification performance. Goli *et al.* [5] took advantages of neighboring vehicles' cooperations to evaluate and predict future vehicle location. Vlahogianni *et al.* [6] realized short-term traffic flow prediction based neural network and proposed a genetic algorithm for proper characteristics representation. Simon *et al.* [7] proposed a model-based approach for short-term travel-time prediction on highway, which is used to on-line and real-time applications and hence. These existing algorithms provide predictive localization but they are mostly short-term prediction results based on real-time driving conditions, such as vehicle speed, road curvature, front congestion, etc.

In fact, long-term predictive localization is particularly valuable in road construction and transportation planning. Even though there have been some long-term predictive localization methods, such as the 60 minutes prediction by Lint [8], the technique of using prior visual memory [9], and DNN-based long-term trajectory prediction [10]. Instead, we try to predict vehicle location distribution days, weeks, or months in the future. Our approach is to separate the problem into two sub-problems: 1) a vehicle clustering sub-problem, which concerns forming stable vehicle clusters based on the behavioral correlation between vehicles. 2) a predictive localization sub-problem, predicting the location of vehicles for long-term future. In this direction, the main contributions of this paper are summarized as follows:

1) We design a Behavior-based Clustering Method (BCM) method to analyze the behavioral correlation between vehicles in each time interval and divide vehicles into non-overlapping clusters. Vehicles with greater behavioral correlation are classified into the same cluster.
2) We propose a Clustering-learning-based Predictive Localization (CLPL) algorithm for IoCV to predict vehicle distribution in the future. By utilizing the BCM results and a deep learning model, CLPL provides long-term prediction without real-time driving conditions.
3) We perform extensive simulations to evaluate the proposed methods. BCM is compared with Sociological Pattern Clustering (SPC) and Mobility-Prediction Based Clustering (MPBC) in terms of clustering performance. Moreover, the matching rate of localization of CLPL

Manuscript received ; revised. Corresponding author: Kai Lin (email: link@dlut.edu.cn).

Kai Lin and Yihui Li are with the School of Computer Science and Technology, Dalian University of Technology, Dalian, China (e-mail: link@dlut.edu.cn, and 928982000@mail.dlut.edu.cn).

J. Deng is with the Department of Computer Science, University of North Carolina at Greensboro, Greensboro, NC 27412 USA (e-mail: jing.deng@uncg.edu).

Pasquale Pace and Giancarlo Fortino are with the Department of Informatics, Modeling, Electronics, and Systems (DIMES), University of Calabria, Italy (email: p.pace@dimes.unical.it, and g.fortino@unical.it).

is compared with reference approaches such as Co-operative Vehicle Localization Algorithm (CVLA) and neighbor-aided localization (NAL). The experiment results show the superiority of our designs in clustering and localization in all the tested scenarios.

The remainder of this paper is organized as follows: Section II introduces related work. Section III presents the system model and the problem formulation. Section IV describes the clustering method based on vehicle behavior detailing the proposed BCM while the novel CLPL algorithm is detailed in Section V. Performance analysis of the presented methods, based on extensive experiments, is conducted in Section VI in which the obtained results are discussed. Finally, Section VII concludes the paper.

II. Related Work

We review recent literature and related works in three different perspectives that are closely related to our work: *A)* vehicle predictive localization, *B)* clustering techniques, and *C)* deep learning support.

A. Vehicle Predictive Localization

Vehicle predictive localization is essential for almost all transportation application services, such as vehicle route selection and traffic congestion reduction. Other than the few we briefly discussed in Section I, there are other important works on predictive localization as well. In terms of short-term prediction, Papakostas and Katsaros proposed a Rich-Dictionary Markov Predictor (RDM) to perform online vehicle next-location prediction with pattern matching [11]. Madhavan and Schlenoff [12] developed a Prediction in Dynamic Environments framework to perform unmanned ground vehicle prediction, which shows the need for sufficiently adequate process models and their importance in short-term moving object prediction. Liu [13] proposed a short-term network traffic forecasting algorithm based on Chaos Theory and Support Vector Machine. Regarding the long-term predictive localization, Hou and Li [14] proposed a long-term traffic flow prediction method based on the repeatability and the similarity of the traffic flow series, which regards more than one day as the long-term prediction period. Su *et al.* [15] proposed a long-term traffic situation prediction model based on functional nonparametric regression, which predicates vehicle location after one day.

However, these existing algorithms, regardless of both long-term and short-term predictive localization, still need improvements in terms of accuracy, scalability, convergence, and effectiveness when facing a large-scale IoCV. In particular, most of these algorithms are only well suited for predictive localization up to 18 days in the future [16], [17], [18]; instead, we focus on more effective predictive localization for a father future in this work, which considers: *Based on past history, how many vehicles will be located at a certain road segment at the specific time of one day, one week, one month or even further in the future? What is the distribution of vehicles' locations within the metropolitan area during that time?*

B. Clustering Techniques

Clustering characteristic provides incredible support for IoCV and many valuable clustering methods were proposed in the past decade [19], [20], [21]. Yang and Leskovec [22] presented an overlapping clustering detection method to detect dense overlapping and hierarchical nest in massive networks. Zhang *et al.* [23] proposed a method called bounded nonnegative matrix tri-factorization (BNMTF) for community detection and addressed the sparsity problem as a result of missing edges in BNMTF. Raghavan *et al.* [24] presented a simple propagation algorithm, which only utilizes the network structure as guidance and requires neither a predefined objective function nor prior information about a cluster. Gregory [25] extended the label and propagation step to include information about more than one cluster. Moreover, vehicle behavior has also been used to classify vehicles in IoCV. Maglaras and Katsaros [26] presented Sociological Pattern Clustering (SPC) and Path Stability Clustering (PSC) to exploit the historic trajectories of vehicles and use clustering primitive of virtual forces to create stable and meaningful clusters. Ni *et al.* [27] proposed a mobility prediction-based clustering (MPBC) method, which adopted mobility prediction strategies to handle various problems caused by node movements. Mattern *et al.* [28] relied on vehicle-to-vehicle (V2V) communication within scope of the cluster to improve positioning accuracy in the Co-operative Vehicle Localization Algorithm (CVLA). Cruz *et al.* [29] designed a Neighbor-Aided Localization (NAL) technique using neighboring vehicles from the same cluster and smartphone inertial sensors for localization prediction. However, the time and space complexity of these algorithms becomes unmanageable when working on large-scale IoCV.

C. Deep Learning Support

Deep learning is another interesting research topic related to our work. In this context, Ravì *et al.* [30] proposed a deep learning methodology that combines features learned from inertial sensor data with complementary information from a set of shallow features to enable accurate and real-time activity classification. It aims to overcome some limitations in typical deep learning framework where on-node computation is required. Wu *et al.* [31] studied leveraging both weakly labeled images and unlabeled images for multi-label image annotation and proposed an approach called weakly semi-supervised deep learning for multi-label image annotation. A weighted pairwise ranking loss is utilized to handle weakly labeled images, while a triplet similarity loss function is employed to harness unlabeled images. In addition, deep learning has been used to beat records in fault diagnosis of electric motors [32], speech recognition [33], emotion detection [34], and intelligent transportation [35], and predict the effects of mutations in non-coding DNA on gene expression and disease [36]. These techniques can certainly be used in the context of long-term predictive localization to ensure the speed and accuracy of the prediction, but need to be improved in combination with IoCV characteristics.

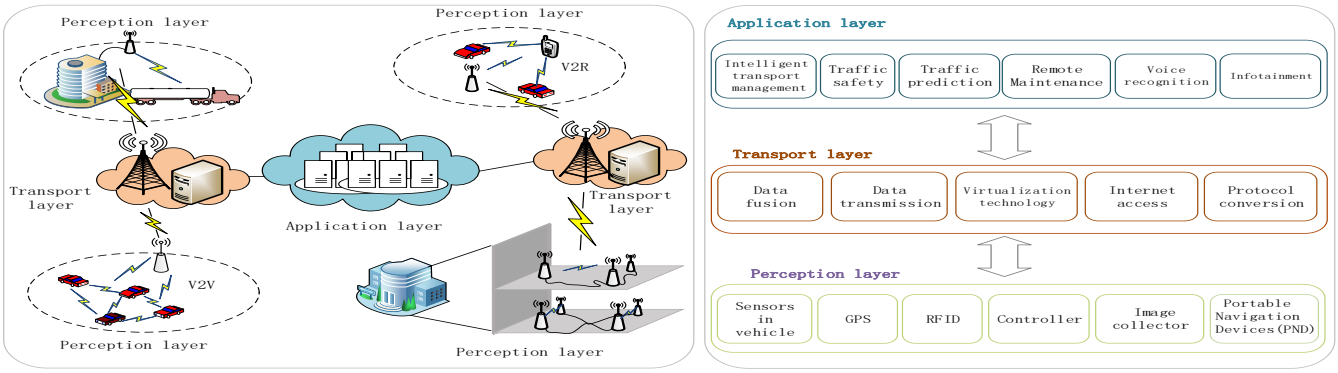


Fig. 1: IoCV system architecture

III. System Model and Problem Formulation

In this paper, we assume that IoCV consists of millions of vehicles, its infrastructure includes numerous RSUs and 5G based stations, and adopts a data center that supports cloud services of big data computing and storage. Cloud service is responsible of storing the large amount of historical vehicle data. Predictive localization will also be performed there. In practical applications, RSUs will occasionally submit newly collected vehicle data to cloud by 5G wireless network in order to enrich the dataset already stored on cloud. Connecting to cloud is only necessary for further data enrichment while vehicles or RSUs can also get online services in this way. Although failed connection could cause loss of some training data and degradation of immediate predictive performance, data resubmissions can be used to address such temporary cloud failures. As shown in Fig. 1, the IoCV system architecture contains three layers: perception, transport, and application, each with different functionalities [37]. Both BCM and CLPL will reside in the application layer.

A. Problem formulation

Long-term predictive localization provides valuable future vehicle distribution information for traffic control, road construction planning, intelligent transportation services in IoCV. Although vehicle's various sensors and intelligent navigation system are expected to collect a rich set of information for predictive localization, such information is easily affected by weather, traffic accidents and other factors, leading to higher prediction errors. Compared with short-term predictive localization, long-term predictive localization is even more difficult to achieve in large-scale IoCV due to error accumulation. In this work, we focus on the following problems of vehicle predictive localization: *Based on available vehicle historical data, how to form stable cluster structures of large-scale vehicles with different states in order to facilitate long-term predictive localization? And how to improve accuracy in the presence of different errors?*

In order to solve the above problems, we analyze historical information of vehicles to classify vehicles with similar behavior. For example, some vehicles often travel and stop at the same place. As an overview, our approach is to first classify vehicles into clusters by mining time-varying behavioral correlation between vehicles. Then, we take advantage of clustering-learning to predict vehicles' long-term location distribution.

B. System model and notations

The following notations are used in this work:

$V = \{v_1, v_2, ..., v_n\}$ represents the vehicle set, $R = \{r_1, r_2, ..., r_u\}$ means the RSUs set.

All roads in IoCV are divided into consecutive road segments for evaluating behavior correlation degree between vehicles to form vehicle clusters, and $S = \{s_1, s_2, ..., s_{card(S)}\}$ represents the set of road segments. Each road segment is served by at least one RSU (Fig. 2), $card(S)$ represents the length of the set S. The RSUs and road segments passed by vehicle v_i at different time-slots are denoted as sets $R_i = \{r(t_1), r(t_2), ..., r(t_{card(R_i)})\}$ and $S_i = \{s(t_1), s(t_2), ..., s(t_{card(S_i)})\}$. In this paper, the predicted period and its corresponding historical period are divided into intervals, and the interval means the time-slot.

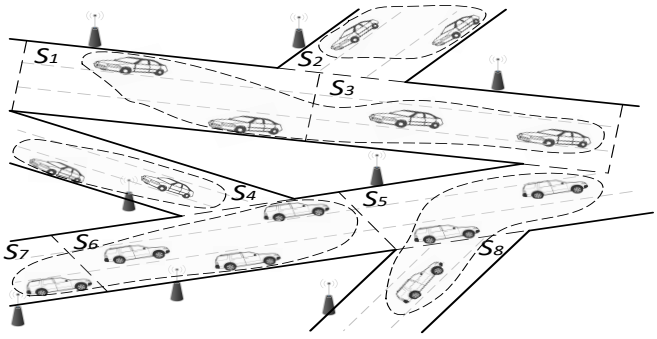


Fig. 2: Road segment division and cluster formation of traffic roads

ψ_{ij} is the behavioral correlation degree between vehicles v_i and v_j belonging to cluster C_i and C_j respectively. Such a degree is measured by the records of their simultaneous driving or parking on the same road segment.

Long-term predictive localization aims to predict vehicle location distribution at any moment in the future, which may

be one day, one week, one month or even further in the future. In this paper, long-term predictive localization is implemented through the deep clustering-learning process on the behavior analysis of the vehicles.

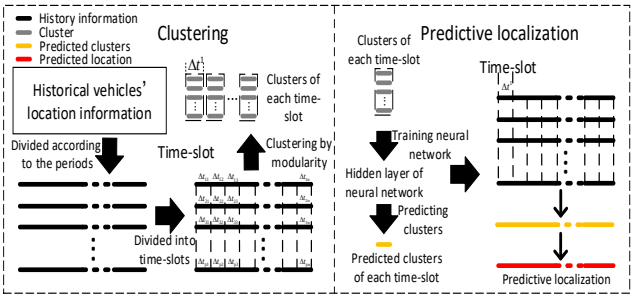


Fig. 3: Implementation processes of long-term predictive localization

As shown in Fig. 3, the implementation process consists of clustering stage and predictive localization stage. Firstly, time period is defined for vehicle behavior analysis, such as one hour, one day, one week, and so on. Then time period is further divided into time-slots as minimum processing units while the vehicle historical data is divided into smaller parts according to the time-slot. The time-slot duration is preset before training, but it can be adjusted according to system capabilities and application requirements. In each time-slot, the clustering stage is executed to form stable vehicle cluster structure by calculating the vehicle behavior correlation and enhancing network modularity. The predictive localization stage uses clustering results of historical corresponding time-slots as input of an improved deep neural network to obtain the cluster structure of a future time-slot, which is then converted to the information of road segments for vehicle predictive localization. For example, in order to predict vehicle location at time t $(t \epsilon \Delta t_{dw})$, clustering results of historical time-slots $\{\Delta t_{1w}, \Delta t_{2w}, ..., \Delta t_{pw}\}(p < d)$ are obtained in the clustering stage and input to the deep neural network for training the cluster structure of time-slot Δt_{dw}, then convert it to the prediction result at time t.

In order to realize long-term predictive localization, the predicted locations are also used as historical information as well as known actual locations to predict the subsequent periods. In addition, the predicted dataset can be corrected with the newly obtained actual locations for eliminating accumulated predicted errors to make the subsequent prediction more accurate.

IV. Clustering Method Based on Vehicle Behavior

Due to the uncertainty of information in IoCV such as the initial distribution of the vehicle and the number of clusters, this section designs a behavior-based clustering method (BCM) as the solution of the first sub-problem (vehicle clustering) instead of selecting existing general clustering methods; moreover, modularity Q is used to enhance BCM's performance.

A. Extraction and Analysis of Vehicle Behavior

In IoCV, vehicle-to-vehicle and vehicle-to-RSUs communications require specific status information such as the IDs of the RSUs or road segments that vehicle v_i passed by, and all the vehicles that communicated with vehicle v_i. In particular, all the IDs of the RSUs passed by v_i during time-slot Δt are firstly extracted from the application layer and stored in the sorted set R_i. Then, the transition from R_i to S_i needs to be accomplished before analyzing vehicle behavior. The following method is adopted to analyze the behavioral correlation degree ψ_{ij} once S_i is determined.

Assuming that the acquisition times are the same for all vehicles, which are equivalent to the number of elapsed time-slots. For any two vehicles v_i and v_j, their sorted sets are denoted as $S_i = \{s(\Delta t_1^i), s(\Delta t_2^i), ..., s(\Delta t_\alpha^i), ..., s(\Delta t_{card(S_i)}^i)\}$ and $S_j = \{s(\Delta t_1^j), s(\Delta t_2^j), ..., s(\Delta t_\alpha^j), ..., s(\Delta t_{card(S_j)}^j)\}$ respectively. The two sorted sets are converted to the corresponding ordered strings $\hat{S}_j$ = "$s(\Delta t_1^i)...s(\Delta t_\alpha^i)...s(\Delta t_{|\hat{S}_i|}^i)$" and $\hat{S}_j$ = "$s(\Delta t_1^j)...s(\Delta t_\alpha^j)...s(\Delta t_{|\hat{S}_j|}^j)$", where $|\hat{S}_i| = card(S_i)$ and $|\hat{S}_j| = card(S_j)$ represent the length of string $\hat{S}_i$ and $\hat{S}_j$ respectively, which also indicate the total number of passed or parked road segments of v_i and v_j during the time-slot Δt, and are used to calculate string similarity for the behavior correlation degree ψ_{ij}.

Existing algorithms calculating string similarity are mainly based on Levenshtein Distance (LD), also called Edit Distance. LD represents the minimum number of operations (insert, delete, replace) needed to convert original string into target string. In particular, Zhang *et al.* combined LD with longest common subsequence (LCS, defined as the longest subsequence of two or more known sequences) between two strings to calculate string similarity, which can trace back all matching paths to improve the accuracy of calculation results [38]. As an extension, we compute string similarity by introducing vehicle characteristic (such as vehicles ID, Time, Longitude and Latitude) into the method described in [38].

For $s_\omega \in$ string $\hat{S}_i$ and $s_\sigma \in$ string $\hat{S}_j$, a matrix $D_{(|\hat{S}_i|)\times(|\hat{S}_j|)} = \{d_{\omega,\sigma}\}$ is built based on the strings $\hat{S}_i$ and $\hat{S}_j$ for calculating the similarity between two vehicles. The value of each element is set according to the following rules: if $s_\omega = s_\sigma (\omega = 0, 1, ..., |\hat{S}_i| - 2$ and $\sigma = 0, 1, ..., |\hat{S}_j| - 2)$, $d_{\omega,\sigma} = 1$; otherwise, $d_{\omega,\sigma} = 0$.

The LCS of two strings represents the number of elements with $d_{(\omega,\sigma)} = 1$ on the backtracking path, which is represented as lcs. A backtracking path in matrix D starts from the bottom right corner element $d_{(|\hat{S}_i|,|\hat{S}_j|)}$ to the upper left corner. The rules of the path backtracking is the following: if the value of current element is 1, moving to the upper or left element; otherwise, moving to the upper, left or upper left element. After finding all the backtracking paths, the one containing the most elements equal to 1 is selected to calculate the lcs.

Let $block(\omega, \sigma)$ represents a rectangle area with the line from the element $d_{((\omega+1),(\sigma+1))}$ to the bottom right corner element $d_{(|\hat{S}_i|,|\hat{S}_j|)}$ as the diagonal. $J(\omega, \sigma)$ is the length of LCS between two strings that are represented by the first column and row of $block(\omega, \sigma)$ without $d_{(\omega,\sigma)}$, respectively.

It is calculated iteratively starting from $d_{(|\hat{S}_i|,|\hat{S}_j|)}$ according to the following rules: if $J(\omega,\sigma+1) \leq J(\omega+1,\sigma)$, $J(\omega,\sigma) = J(\omega,\sigma)$; otherwise, $J(\omega,\sigma) = J(\omega,\sigma+1)+1$ when $d_{\omega,\sigma+1} \neq 1$, $J(\omega,\sigma) = J(\omega,\sigma+2)+1$ when $d_{\omega,\sigma+1} = 1$. $J(0,0)$ is simply LD, denoted as ld, of strings $|\hat{S}_i|$ and $|\hat{S}_j|$.

With the obtained ld and lcs from the above analysis, the behavioral correlation between v_i and v_j (also represented by the similarity between $\hat{S}_i$ and $\hat{S}_j$) is calculated as follows:

$$\psi_{ij} = \frac{lcs}{ld + lcs + \frac{|\hat{S}_i| - \varrho}{|\hat{S}_i|}} \tag{1}$$

where ϱ is the subscript of the element that changes the first value on the diagonal from the upper left corner in the matrix $D_{(|\hat{S}_i|)\times(|\hat{S}_j|)}$ representing the common prefix for string $\hat{S}_i$ and $\hat{S}_j$.

In addition, the relevance $\psi_{i'j'}$ between two clusters $C_{i'}$ and $C_{j'}$ is calculated based on the behavioral correlation between vehicles from different clusters:

$$\psi_{i'j'} = \frac{1}{card(C_{i'}) \cdot card(C_{j'})} \sum_{l=1}^{card(C_{i'})} \sum_{k=1}^{card(C_{j'})} \psi_{lk} \tag{2}$$

where $v_l \in C_{i'}$, $v_k \in C_{j'}$, $C_{i'}$ and $C_{i'}$ include vehicle $v_{i'}$ and $v_{j'}$, respectively.

B. Behavior-based Clustering Method

BCM classifies vehicles into the cluster set $C = \{C_1, C_2, ...C_{i'}, ..., C_{card(C)}\}$. In the initial stage of clustering process, each vehicle is treated as an initial cluster and is continuously merged in the subsequent process. In order to clearly describe the relevance between two formed clusters, we define an adjacency matrix as follows:

$$\Psi_{q\times q} = \begin{pmatrix} \psi_{11} & \cdots & \psi_{1q} \\ \vdots & \ddots & \vdots \\ \psi_{q1} & \cdots & \psi_{qq} \end{pmatrix} \tag{3}$$

where q represents the number of clusters.

Each element in the matrix represents the relevance between two clusters. Each element on the main diagonal indicates the relevance between each cluster and itself and is set to zero. Moreover, in order to further reduce the time and space complexity of the clustering process, as well as avoiding unnecessary operations, a preset threshold T_r ($T_r \geq 0$) can be introduced while every other element in $\Psi_{q\times q}$ is set to zero if its value is less than T_r. This threshold affects the size of the adjacency matrix for cluster formation and its value is related to each specific application scenario. A larger T_r means a higher probability of setting elements in to zero, i.e., the possibility of merging different clusters is lower, which increases the size of adjacency matrix and leads to, on an average, more clusters are ultimately formed while fewer road segments and vehicles are associated with each cluster. Although introducing T_r reduces the complexity, it comes at the expense of a slight reduction in accuracy, so it is necessary to decide whether to use it depending on the actual application.

The clustering structure in BCM is updated on a time-slot interval Δt, in which the cluster set C and the adjacency matrix $\Psi_{q\times q}$ are initialized. C_h represents the initial cluster set of the h^{th} iteration in Δt; then, the next step is to search the maximum element in the adjacency matrix to find the corresponding two clusters $C_{i'}$ and $C_{j'}$ that can be merged into a new cluster C_η ($C_\eta = C_{i'} \cup C_{j'}$).

Modularity Q measure [39] is widely used in evaluating optimization methods to measure the strength of division of a network into modules (also called groups, clusters or communities). Such a measure usually reflects the tightness of node connections within the same module and the sparsity of node connections between modules. In this paper, nodes and edges represent vehicles and the existence of the behavioral correlation between them, respectively. The maximization of network modularity is pursued to obtain a stable cluster structure with high behavior correlation between vehicles.

As discussed in Section IV.A, the behavioral correlation between vehicles is calculated by the common passed or parked road segments of vehicles during the time-slot Δt. In our BCM scheme, each vehicle is treated as one cluster in the initialization phase. A greedy algorithm is used to merge the most relevant clusters (vehicles) during the initial stage without sufficient historical data for training, which makes clustering processing more efficiency and less time-consuming. Through iterations, different vehicles and road segments will be merged into clusters, improving the tightness of our approach. For example, in h^{th} iteration, the clusters $C_{i'}$ and $C_{j'}$ are merged as a C_η if such a merge would increase modularity Q. Assuming that the merge of these two clusters is performed, the updated modularity is calculated as follows (h = 1, 2, ...):

$$Q_h = \sum_i (e_{ii} - a_i^2) = \sum_i e_{ii} - \sum_i a_i^2 = T_r(e) - \left\|e^2\right\| \tag{4}$$

where e is a symmetric matrix whose element e_{ij} represents the fraction of all edges in the network that link vehicles in cluster i to those in cluster j. $T_r(e) = \sum_i e_{ii}$ gives the fraction of edges in the network that connect vehicles in the same cluster. The row (or column) sums $a_i = \sum_j e_{ij}$ indicates the fraction of edges that connect to vehicles in cluster i. $\left\|e^2\right\| = \sum_i (\sum_j e_{ij})^2$ represents the sum of all elements in matrix e^2 [40].

If $Q_h^{'} < Q_h$, the clusters $C_{i'}$ and $C_{j'}$ are not merged, BCM terminates because modularity Q cannot be increased by merging the clusters with highest similarity. Otherwise, the cluster $C_{i'}$ and cluster $C_{j'}$ are merged into a new cluster C_η, and the clustering set changes from C_h to $C_h^{'}$. In addition, considering the situation that $Q_h^{'} - Q_h \leq \varepsilon$ ($\varepsilon \geq 0$ is a small preset value), continuing to execute iterations can only lead to a small increase in modularity at most, BCM is also terminated to reduce overhead. In this case, $C_{h_f}^{'}$ is the final clustering set and the modularity Q is equal to $Q_h^{'}$. If $Q_h^{'} - Q_h > \varepsilon$, BCM continues the iterative process to take advantage of the increase in modularity, where $Q_{h+1} = Q_h^{'}$ and $C_{(h+1)} = C_h^{'}$. Similar to T_r, ε is introduced to expedite algorithm convergence,

below which clusters merging will not take place. This is a simple mechanism to reduce computation cost with the expense of accuracy and we leave further investigation of the impact of T_r and ε to our future work. The pseudocode of BCM is summarized in Algorithm 1.

Algorithm 1 - BCM working sequence

Input: The initialized clustering result set C of each time-slot.

Output: Solution to the first sub-problem, i.e. vehicle clustering result of each time-slot.

Initializing $\Psi_{q\times q}$.
outer:
for each time-slot Δt **do**
  for each iteration h **do**
    Search for the maximum element in the adjacency matrix and obtain the corresponding two clusters $C_{i'}$ and $C_{j'}$.
    Generate a new cluster $C_\eta = C_{i'} \cup C_{j'}$, update the clustering result set $C_{(h+1)}$ and calculate the modularity measure $Q_h^{'}$
    if $Q_h^{'} < Q_h$
      The algorithm terminates.
      The clusters $C_{i'}$ and $C_{j'}$ are not merged
    end if
    else if $Q_h^{'} - Q_h > \varepsilon$
        $Q_{h+1} = Q_h^{'}$
        $C_{(h+1)} = C_h^{'}$
        Re-build the adjacent matrix $\Psi_{|C_{(h+1)}|\times|C_{(h+1)}|}$
      Continue **outer:**
      end if
      else
        The algorithm terminates.
        $C_h^{'}$ is the final clustering set
        The modularity Q is equal to $Q_h^{'}$
      end else
    end else
  end for
end for

V. Long-term predictive localization scheme

In this section, a long-term predictive localization scheme is proposed to solve the second sub-problem (predictive localization) by using an improved deep learning method. The BCM result is used as the input of deep neural network.

A. Vehicle Cluster Behavior Analysis

Considering two consecutive time-slots $\Delta t-1$ and Δt, the clusters formed by BCM in IoCV are denoted as $C(\Delta t-1) = \{C_1(\Delta t-1), C_2(\Delta t-1), ..., C_{|C(\Delta t-1)|}(\Delta t-1)\}$ and $C(\Delta t) = \{C_1(\Delta t), C_2(\Delta t), ..., C_{|C(\Delta t)|}(\Delta t)\}$, respectively. If the road segment information of all vehicles in clusters $i^{'}$ are not changed and no new vehicles are added, the number and IDs of vehicles in cluster $i^{'}$ do not change from time-slot $\Delta t-1$ to Δt, thus holding $C_{i'}(\Delta t-1) = C_{i'}(\Delta t)$. If multiple similar behaviors are merged, the corresponding multiple small-scale clusters are merged into a larger cluster.

The change of vehicle clusters is reflected through $C(\Delta t-1)$ and $C(\Delta t)$, which cannot be expressed by a linear structure; thus, CLPL based on deep learning for such non-linear structure, is described in the next subsection.

B. Clustering-learning-based Predictive Localization

In each time-slot Δt, the cluster structure is formed based on the vehicle behavior correlation, which includes various characteristic information, such as time characteristic (the form and decomposition time of cluster structure), vehicle characteristic (ID, Longitude, Latitude, road segment), cluster characteristic (the number of clusters, the number of vehicles in each cluster, the coverage of cluster), etc. The model SAE [41] has the ability of realizing high-dimensional feature extraction and clustering learning with fast convergence and high accuracy. Thus, we propose an Improved Stacked Auto-Encoder (ISAE) model to predict the cluster structure of vehicles in the future. As shown in Fig. 4, the ISAE model includes multiple original SAE layers and one prediction layer on top. Clustering results of historical corresponding time-slots are treated as the initial input of SAE layers for training data features associated with vehicle clusters. The introduced prediction layer uses the refined cluster feature data from the adjacent SAE layer to predict the future cluster structure, which is converted to the predicted location based on the mapping relationship between clusters and road segments. The vehicle distribution at any time is obtained from the predicted locations of all vehicles. The training sample set is denoted as $\{x_1, x_2, x_3, ...x_\ell, ..., x_N\}$, where N is the number of samples and each sample contains cluster structure information of one time-slot (the cluster structure is identified by BCM, as discussed in Section IV).

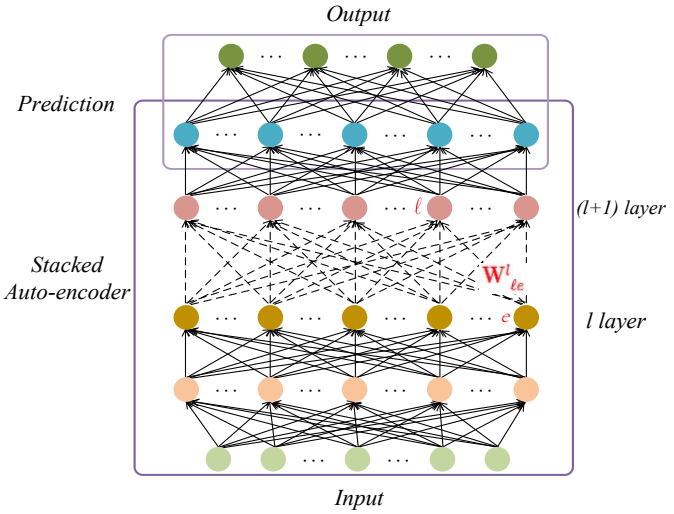


Fig. 4: Improved stacked auto-encoder model

Before introducing ISAE, we present encoding and decoding functions as follows:

$$y(x_\ell) = f(\mathbf{W_1} x_\ell + \mathbf{b}) \tag{5}$$

$$z(x_\ell) = g(\mathbf{W_2}y(x_\ell) + \mathbf{c}) \tag{6}$$

where $\mathbf{W_1}$ and $\mathbf{W_2}$ are the encoding and decoding weight matrixes, $\mathbf{b}$ and $\mathbf{c}$ are the encoding and decoding bias vectors respectively. ISAE is trained in a greedy layer-wise way. Given a set of training samples $\{x_1, x_2, x_3, ...x_\ell, ..., x_N\}$, where $x_\ell \in R^d$, an auto-encoder first encodes an input x_ℓ to a hidden representation $y(x_\ell)$ based on equation (5), and then it decodes representation $y(x_\ell)$ back into a reconstruction $z(x_\ell)$ according to equation (6). The $sigmoid$ function is used as the transfer function for the process of encoding and decoding, which is defined as follows:

$$f(x) = g(x) = \frac{1}{1 + \exp(-x)} \tag{7}$$

As shown in Fig. 5, during the training process of each original SAE layer, the weight matrix W_1, W_2, and bias vector b_1, b_2 in the layer are obtained by encoding and decoding the input dataset, and the hidden layer of this training process is used as input for the next layer. The same strategy is carried out for the following layers to initialize the weight matrix of the network and an overall cost function H to evaluate the weight matrix and bias vector of the input auto-encoder in each layer. The cost function is defined as follows:

$$H = L(X, Z) + \delta\ U(\rho||\rho_e) \tag{8}$$

$L(X, Z)$ is the specific cost function and $U(\rho||\rho_e)$ is a weighted penalty term.

$$L(X, Z) = \underset{X,W\in(0,1)}{\arg\min} \frac{1}{2}\sum_{l=1}^{N} \|x_\ell - z(x_\ell)\|^2 + \frac{\lambda}{2} J_{\mathbf{W}^l} \tag{9}$$

where X and W are the finite set of x_ℓ and $\mathbf{W}^l$; Z represents the finite set of $z(x_\ell)$ while λ represent the weight parameter for the weight matrix. The $J_{\mathbf{W}^l}$ term is defined as follows:

$$J_{\mathbf{W}^l} = \sum_{l=1}^{n_l-1} \sum_{j=1}^{\hat{s}_l} \sum_{e=1}^{\hat{s}_{l+1}} \left(\mathbf{W}_{je}^l\right)^2 \tag{10}$$

where $\mathbf{W}_{je}^l$ indicates the weight between the e^{th} neuron on the layer l and the j^{th} neuron on the layer $(l+1)$ while $\hat{s}_l$ and $\hat{s}_{l+1}$ represent the number of neuron nodes in two adjacent layers, respectively. N_l represents the number of input samples for layer l.

The weighted penalty term is defined as follows:

$$U(\rho||\rho_e) = \rho \ln \frac{\rho}{\rho_e} + (1-\rho) \ln \frac{1-\rho}{1-\rho_e} \tag{11}$$

The average activity of the generic neuron e can be expressed as:

$$\rho_e = \frac{1}{N}\sum_{\ell=1}^{N} y_e(x_\ell) \tag{12}$$

where x_ℓ is the input, $y_e(x_\ell)$ is an activity for the neuron e, and N is the number of input samples. We used the Kullback-Leibler divergence as a regular constraint of the network to make ρ_e as close as possible to a decimal value ρ that approaches zero.

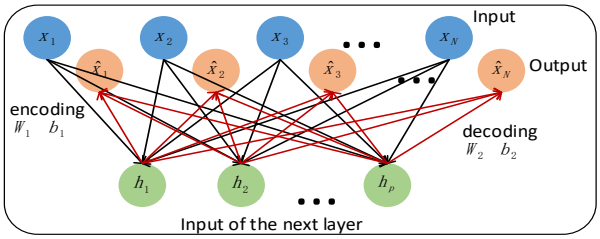


Fig. 5: The architecture of stacked auto-encoder model

Through the above analysis, a weight matrix $\mathbf{W}$ and a bias vector $\mathbf{b}$ can be obtained after an auto-encoding process. For an inter-layer process, the output is obtained through a single layer model with raw data, where the cost function represents the system predictive performance. In this context, we used Limited-memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS) [42] to update the bias vector and weight matrix during the cost function minimizing process, which provides generalization bounds for linear and nonlinear kernels and has high learning performance.

The updated rules are the followings:

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \mathbf{A}_k \frac{\partial H}{\partial \mathbf{W}^m} \tag{13}$$

$$\mathbf{b}^m(k+1) = \mathbf{b}^m(k) - \mathbf{A}_k \frac{\partial H}{\partial \mathbf{b}^m} \tag{14}$$

where $\mathbf{A}_k$ is the inverse of the Hessian matrix and the partial derivative of a weight matrix and a bias vector are calculated as:

$$\frac{\partial H}{\partial \mathbf{W}_{\ell e}^m} = \frac{\partial H}{\partial f_m}\frac{\partial f_m}{\partial \mathbf{W}_{\ell e}^m} = -\delta_e^m \cdot z'_{(m-1)e} + \lambda \mathbf{W}_{\ell e}^m \tag{15}$$

$$\frac{\partial H}{\partial \mathbf{b}_e^m} = \frac{\partial H}{\partial f_m}\frac{\partial f_m}{\partial \mathbf{b}_e^m} = -(z_{m\ell} - z'_{m\ell}) f'_m = -\delta_e^m \tag{16}$$

where m represents the m^{th} layer. f_m is the $sigmoid$ function of the m^{th} layer which is calculated through equation (7). δ_e^m represents the residual of the neuron e on the m^{th} layer while $z'_{m\ell}$ is the derivative of the decoding function of the neuron e on the m^{th} layer and $z_{m\ell}$ is calculated through equation (6).

During the training process, CLPL uses a greedy layer-wise way to prevent the occurrence of gradient diffusion, which contains an initialization phase and fine-tuning steps for model parameters as shown in Fig. 6.

In particular, within the initialization stage of CLPL, the original data is pre-processed and both the weight matrix $\mathbf{W}$ and the bias vector $\mathbf{b}$ are initialized. Then a hidden layer is trained to determine whether the cost function H is minimized. $\mathbf{W}$ and $\mathbf{b}$ are updated according to the LBFGS algorithm for the training in the next hidden layer. Once achieved the minimization of the cost function H, if the last layer of the training process is reached, the initialization stage is

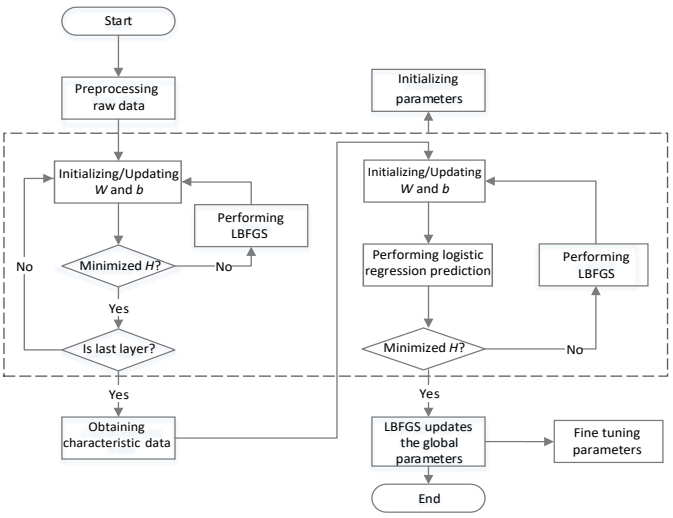


Fig. 6: Training process of CLPL

terminated and the characteristic data is obtained; otherwise, $\mathbf{W}$ and $\mathbf{b}$ need to be computed through the next layer.

According to the above analysis, the prediction result at time-slot $\Delta t + \Delta T$ is the formed vehicle cluster structure, which is denoted as follows:

$$C(\Delta t+\Delta T) = \{C_1(\Delta t+\Delta T), C_2(\Delta t+\Delta T), ..., C_q(\Delta t+\Delta T)\} \tag{17}$$

where ΔT represents the sum of all time-slots from current time-slot Δt to the predicted time-slot and q is the number of predicted vehicle clusters at the time-slot $\Delta t+\Delta T$. According to the previous assumption, there is at least one RSU in the geographic range corresponding to each cluster. For simplicity of expression, the smallest cluster covering only one RSU is represented by the RSU's ID as a subscript, and a merged cluster is represented by all the IDs of its covered RSUs as the subscript. For example, two cluster C_{r1} and C_{r2} are merged at the time-slot $\Delta t + \Delta T$ and the formed new cluster expressed as $C_{(r1,r2)}$. The prediction result can be also described with RSU IDs as follows:

$$C^{\Delta t+\Delta T} = \{C^{\Delta t+\Delta T}(r_1, r_2, ..., r_{i'}), C^{\Delta t+\Delta T}(r_{i'+1}, r_{i'+2}, ..., r_{i'+\iota}), ..., C^{\Delta t+\Delta T}(r_{\tau'+1}, r_{\tau'+2}, ..., r_u)\} \tag{18}$$

The overall operational sequence of CLPL is summarized in Algorithm 2.

C. Precise Vehicle Location

Since the training data can be collected by V2V and V2R, CLPL is able to work under the premise of GPS loss due to weak signal or interference to realize the long-term prediction of vehicle location. In other words, although GPS provides the most accurate training data to improve prediction accuracy, it is unnecessary for CLPL all the time. Regarding the precise vehicle location in the road segment, we discuss the following two situations:

1) Current precise location: Regarding the current precise location of each vehicle, we can get accurate results by using GPS or other smart navigation devices. In addition, many existing algorithms are available using different technologies such as the RFID technology through which the generic vehicle v_i exchanges location information with its neighbors at the time-slot Δt; then, the updated location information and signal strength are stored in the following sets:

$$\{i = \{1, 2, \cdots, V_{nb}\} \mid \text{id} : v_i, (lng_i^{\Delta t}, lat_i^{\Delta t})\} \tag{19}$$

$$\{i = \{1, 2, \cdots, V_{nb}\} \mid \text{id} : v_i, (P_i)\} \tag{20}$$

where V_{nb} is the number of vehicles in the neighborhood of vehicle v_i, lng and lat represent the longitude and latitude coordinates, P_i is the received power in dBm. Then the current location of vehicle v_i can be computed.

2) Future precise location: Regarding the future precise location, the existing algorithms mentioned above are able to make short-term predictions, which are beyond the scope of this paper. In contrast, long-term precise location prediction cannot be achieved due to the lack of required environmental information near the prediction time. Therefore, the center of located road segment obtained by long-term prediction of CLPL is considered as the precise location of the vehicle at the prediction time. In this case, the length and width of road segment determines the range of the localization error, which is limited to half the length of the diagonal of the road segment.

Algorithm 2 - CLPL working sequence

Input: Historical behavior information of vehicles and clustering result of first sub-problem at different time-slots.
Output: Vehicle location distribution information at time-slot $\Delta t + \Delta T$.

for each time-slot Δt **do**
  for each vehicle $v_i \in V$ **do**
    Extract the behavior information and store it in set $S_i = \{s(\Delta t_1^i), s(\Delta t_2^i), ..., s(\Delta t_\alpha^i), ..., s(\Delta t_{card(S_i)}^i)\}$.
    Execute the conversion from S_i to $\hat{S}_i =$ "$s(\Delta t_1^i)...s(\Delta t_\alpha^i)...s(\Delta t_{|\hat{S}_i|}^i)$".
  end for
end for
for each time-slot **do**
  $\forall v_i, v_j$, compute ψ_{ij} via (1).
  Initialize $\Psi_{q\times q}$.
  Detect the cluster structure at current time-slot Δt by using BCM.
end for
Use ISAE to train cluster data at different time-slots and compute the weight matrix $\mathbf{W}$ and the bias vector $\mathbf{b}$.
Predict a cluster $C(\Delta t + \Delta T)$ at time-slot $\Delta t + \Delta T$ by using $\mathbf{W}$ and $\mathbf{b}$.
Execute the conversion of predicted results with RSU IDs via (18)
Get the road segment information of all the vehicles located at the specific target prediction time.

D. Complexity analysis of CLPL

Complexity analysis is mandatory to shed light on the actionability of the proposed approach [43], thus the complexity of CLPL is analyzed for each of the two processes: clustering and localization. Supposing each RSU covers an average of V_R vehicles and the presence of R RSUs in the network involved into the clustering process, the maximum number of connections to discover the correlations between clusters is $RV_R(RV_R-1)/2$. In addition, considering the worst case for calculating behavior correlation between each pair of vehicles in the re-clustering phase, the number of inter-cluster correlations does not exceed $RV_R(RV_R-1)/2$, thus, the complexity of CLPL's clustering process is $\mathcal{O}(RV_R(RV_R-1)/2 + RV_R(RV_R-1)/2)$ whose order of magnitude is quadratic and dominated by $\mathcal{O}(RV_R)^2$ where the product RV_R is simply the total number of vehicles in IoCV.

The clustering process of the CLPL scheme is centralized and its complexity increases with the increasing number of vehicles; thus considering that the correlation between vehicles far away is very low, a large area can be divided into sub-regions to improve the scalability of CLPL. For example, the area of a state can be divided according to the boundaries of its cities. Each intra-subregion performs a centralized clustering while different subregions run in a distributed manner.

Regarding the complexity of the localization process of CLPL, the analysis is based on the assumption that f hidden layers are supported by the scheme and each hidden layer have N_h nodes. N samples are adopted to feed-forward calculations and $(f+2)$ matrix operations are performed (*i.e.,* multiplication operation between a vector and a matrix). Since the number of nodes from the input layer to the final output layer (N_{in} and N_{out}) are deterministic and can be considered as constants, the complexity of the feed-forward computation is $\mathcal{O}(N_{in} \cdot N_h + N_h \cdot N_h + ... + N_h \cdot N_{out})$ which can be represented as $\mathcal{O}(N \cdot ((f+2) \cdot N_h^2)$. Moreover, due to the symmetric nature of the process, the complexity of the reverse propagation is the same as the feed-forward. According to the previous considerations, the complexity of the localization process is also quadratic in terms of magnitude order and it is dominated by $\mathcal{O}(N_h)^2$. Finally, we can argue that the overall complexity of CLPL is $\mathcal{O}((RV_R)^2+(N_h)^2)$. It is worth noting that, although the overhead of CLPL increases with the number of vehicles, an algorithm with such complexity can easily be supported by modern high performance computing platforms.

In terms of the time complexity of SPC and MPBC, assuming that the number of vehicles is regarded as N, the time complexity in SPC and MPBC can be represented as $\mathcal{O}(N)$ and $\mathcal{O}(N^2)$.

VI. Experiments and Results

In this section, simulations are carried out to evaluate BCM and CLPL in terms of several performance criteria and the obtained results are discussed in detail.

A. Experimental Dataset

In our experiments, BCM and CLPL have been tested by using real datasets of vehicles of Nanjing and Chengdu, China and LA and Tampa, USA. The data maps and datasets of four areas have been imported in Eclipse software and processed by Eclipse and MongoDB database. All major roads in the region are divided into consecutive road segments with unique IDs. The length of road segments is set as 30m, 50m, 70m and 90m for different evaluations. Every road segment is deployed one RSU. Assume that there are enough 5G base stations to provide seamless link communication services in these four areas. The number of vehicles and the hidden layer are given in TABLE II and III respectively. It is worth noting that all kinds of traffic information are extracted from these real datasets instead of being set. We first used the BCM scheme to correlate vehicle behaviors. Then, the data of the first eleven and a half months is regarded as the input of Fig. 4 for training, which in turn predicts the future locations (output in Fig. 4). The predictions are then validated and compared to the historical data in our data set.

As an illustration, the historical and one-day-later predicted vehicle location of Nanjing at 10:00AM on December 18, 2017 are shown in Fig. 7. In particular, Fig. 7(a) shows the actual vehicle distribution in part of the experimental region based on dataset records while Fig. 7(b) presents the distribution of vehicles predicted by using CLPL with all historical records before 10:00AM on December 17, 2017.

B. Clustering Accuracy of BCM

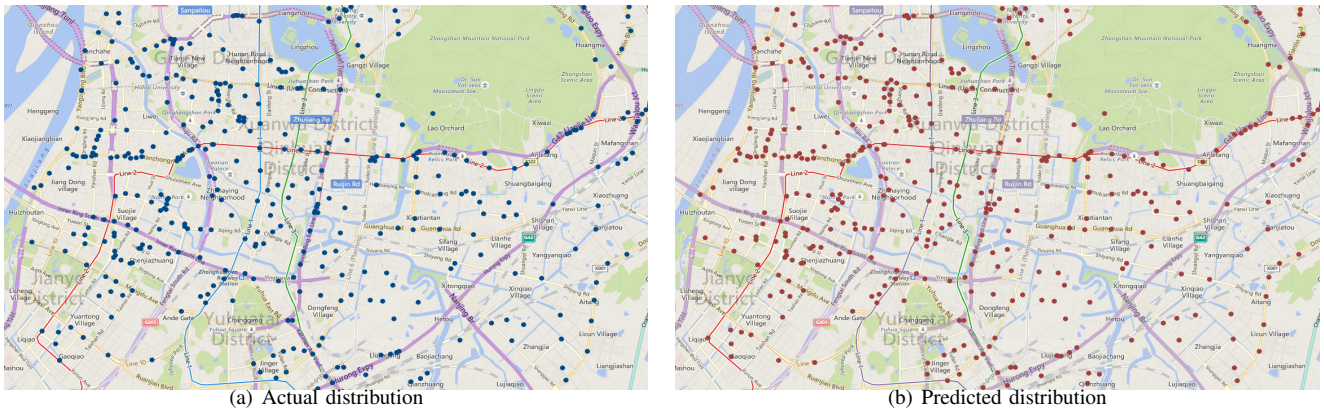
The evaluation of BCM consists of two parts: self-comparison using different parameters and comparison with other two methods, namely SPC [26] and MPBC [27]. Several control parameters have been used: γ is the degree distribution exponent that reflects the probability of forming clusters in the network with average degree k;[1] β is the exponent of cluster size distribution that allows us to adjust cluster size; m_w is the weight parameter of mixed clustering, with m_w of edges connected to the nodes in other clusters (and $1-m_w$ with those within the same cluster); μ is a mixing parameter to represent the mixing degree between different clusters formed in IoCV.

Two performance metrics are used for comparison: Normalized Mutual Information (NMI) and modularity Q. NMI [44] is one of the commonly used indicators for evaluating clustering algorithms, which reflects the similarity between the underlying cluster structure of a network and the cluster structure actually formed by the algorithm. The value of NMI more closer to 1 means better clustering result.

We also adopt modularity Q to quantify a vehicle clustering structure, as in Eq. (4). Edges connected to vehicles in cluster i have different weight values. The weight value is 1 if two vehicles linked by this edge are in the same cluster, otherwise is 0.

NMI results are compared in Fig. 8 for different control parameters. Overall, NMI of all schemes, BCM, SPC, and MPBC lowers as μ increases. BCM can be seen as consistently outperforming SPC and MPBC for about 30%.

[1]The average degree is equal to the sum of the degrees of all nodes divided by twice the number of nodes; the degree of a node is the sum of the indegree and outdegree of the node in the directed graph.

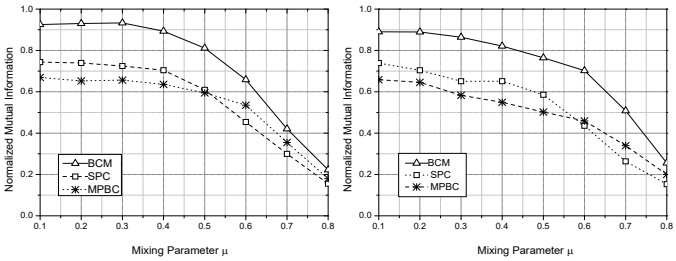


(a) Actual distribution (b) Predicted distribution

Fig. 7: Illustration of vehicle location distribution for 10:00 AM on December 18, 2017. a) Actual distribution; b) Predicted by our schemes based on the training of January 1, 2017 to December 15, 2017

TABLE I: The average size, standard deviations of size, minimum size, and maximum size of clusters during 1250-4000 seconds.

Name	BCM	SPC	MPBC
Nanjing vehicle network	960.17/72.57/1125/900	1140.75/108.94/1320/1000	1025.25/83.14/1208/925
LA vehicle network	430.00/31.21/500/393	437.25/43.94/565/390	437.17/41.19/520/389
Chengdu vehicle network	3234.08/425.04/4250/2831	3417.33/580.07/4490/2710	3473.00/599.49/4612/2635
Tampa vehicle network	115.25/16.51/145/96	116.83/17.88/158/92	117.58/17.60/166/97



(a) $\gamma = 2$, $\beta = 1$, $m_w = 0.1$ (b) $\gamma = 3$, $\beta = 2$, $m_w = 0.5$

Fig. 8: Performance evaluation of BCM with different control parameters

In next experiments, the algorithms have been further discussed in case of $\gamma = 2$, $\beta = 1$, and $m_w = 0.5$. The number of clusters of four vehicle networks for BCM, shown in Fig. 9(a) to (d), started much higher than the other two schemes, but quickly dropped off and stabilized at around 870, 410, 3000, and 105. Instead, both SPC and MPBC saw large fluctuations due to vehicle mobility. In Fig. 9(e) to (h), cluster formation delay increases with μ, due to increasing community structure complexity. BCM is shown to enjoy about 15% shorter delays than SPC, which is about 40% better than MPBC.

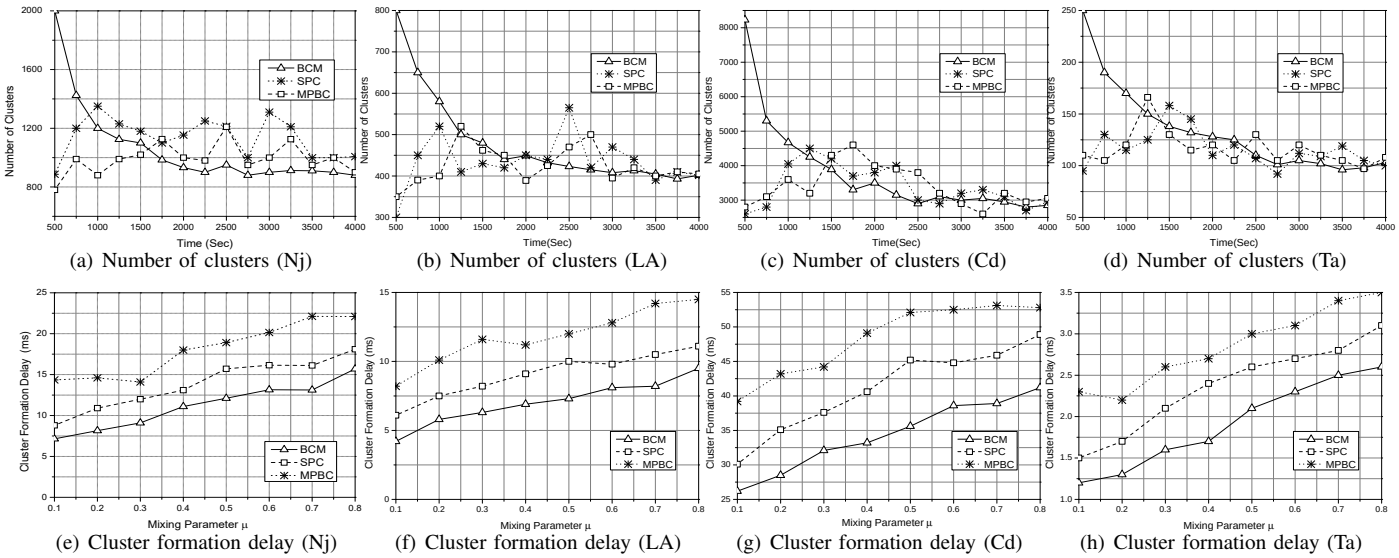
In TABLE I, we show the average size, standard deviations of sizes, minimum size, and maximum size of clusters formed by all compared techniques during 1250-4000 seconds. It can be seen that the BCM scheme has the smallest and most stable result compares to SPC and MPBC. However, it should be noted that TABLE I only shows the stabilized results. As discussed above, BCM assumes each vehicle as one cluster at the beginning of the algorithm execution and its number of clusters decreases quickly, as shown in Fig. 9(a) to (d).

The comparison results of cluster formation delay and modularity Q for BCM, SPC, and MPBC in the four real networks are given in TABLE II. In general, cluster formation delays are longer and modularity Q values are smaller when networks are more complex. BCM is shown with shorter cluster formation delay and better modularity Q for all five real datasets, some of which are complex. Compared with other two techniques, BCM stabilizes faster and obtains fewer clusters, making it suitable for long-term prediction and large-scale network applications.

The time and space complexity of BCM, SPC, and MPBC are analyzed. N_t represents the number of time-slots, BCM has an initial complexity of N^2 in order to compute the adjacency matrix. Then it needs $((N_t - 1)N^2)$ computations in the following N_t time-slots to rebuild the matrix, so it has $O(N_t N^2)$ time complexity. The space complexity of BCM is $O(N^2)$ because only the latest matrix is stored. For SPC, both calculating the social behavior and comparing the states

TABLE II: Cluster formation delay and Modularity Q comparisons for BCM, SPC, and MPBC under the four real datasets

Name	Nodes	Edges	BCM	SPC	MPBC
Nanjing vehicle network	7,690	27,561	2.22/0.42	3.70/0.39	4.20/0.38
Chengdu vehicle network	33,964	103,245	5.35/0.32	6.20/0.29	6.89/0.27
LA vehicle network	3,233	12,325	1.23/0.48	1.62/0.45	2.03/0.43
Tampa vehicle network	873	3,386	0.31/0.55	0.42/0.53	0.51/0.49



(a) Number of clusters (Nj) (b) Number of clusters (LA) (c) Number of clusters (Cd) (d) Number of clusters (Ta)

(e) Cluster formation delay (Nj) (f) Cluster formation delay (LA) (g) Cluster formation delay (Cd) (h) Cluster formation delay (Ta)

Fig. 9: Number of clusters and cluster formation delay comparison. Nj: Nanjing, China. LA: Los Angeles, CA, USA. Cd: Chengdu, China. Ta: Tampa, FL, USA.

among vehicles in all time-slots need $O(N_t N)$ time and $O(N)$ space; therefore, its time complexity is $O(N_t N)$ and space complexity is $O(N)$. MPBC uses $O(N_t N^2)$ time and $O(N)$ space to construct clusters by building the neighbor lists for all vehicles in N_t time-slots, then $O(N_t N)$ time and $O(N_t N)$ space are introduced to judge the stability of clusters, so it has $O(N_t N^2)$ time complexity and $O(N_t N)$ space complexity.

Even though BCM has relatively higher computation complexity and storage requirement than SPC and MPBC, its clustering performance is much superior. We argue that this is necessary for CLPL to achieve its higher performance in all other metrics that we have evaluated in the following subsection.

C. Results of CLPL

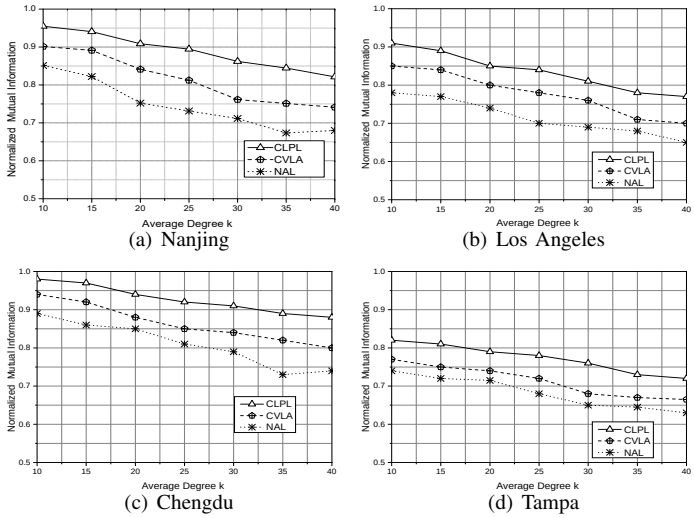
We compared CLPL with two techniques: CVLA [28] and NAL [29]. CVLA incorporates digital maps in map matching and uses vehicle-to-vehicle communication to improve ego positioning. NAL employs a two-stage Bayesian filter to track vehicle locations, without the help of GPS data.

We compare these schemes mainly in terms of matching rate, simply defined as the ratio of vehicles that are predicted to be in the same road segments as they really are and the total number of vehicles that we investigated.

TABLE III: Parameters for different scenarios

ζ (minutes)	Hidden Layer	Hidden Units
1.0	4	[400,400,400,400]
2.0	5	[300,300,300,300,300]
3.0	6	[400,400,400,400,400,400]
4.0	6	[500,500,500,500,500,500]

In order to predict the vehicle location distribution, it is necessary to consider the time-slot duration ζ, the number of hidden layers, and the number of hidden units contained in each hidden layer. Note that ζ is the duration of the time slot, i.e., the duration of the minimum processing time unit rather than the predicted time. In CLPL experiments, the time-slot duration is set from 0.5 minutes to 4.5 minutes, and the number of hidden layers varies from 1 to 8, and the number of hidden units is selected between 100 and $1,000$ with 100 increments. TABLE III summarizes the most suitable architectures in different scenarios. For instance, when $\zeta = 3.0$ minutes, the number of hidden layers is 6 and each layer contains 400 units.



(a) Nanjing (b) Los Angeles

(c) Chengdu (d) Tampa

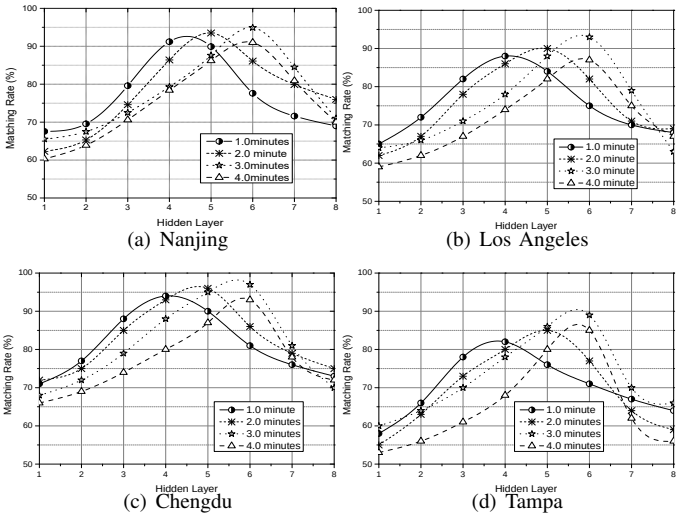
Fig. 10: NMI performance comparison for CLPL, CVLA, and NAL with different average degrees. Note that the results have been plotted starting from 0.5 for better clarity.

In Fig. 10(a) to (d), we compare CLPL, CVLA, and NAL for NMI performance with different average degrees k. CLPL was able to achieve consistently higher NMI compared to CVLA and NAL, mainly thanks to BCM.

Fig. 11(a) to (d) shows how the number of hidden layers

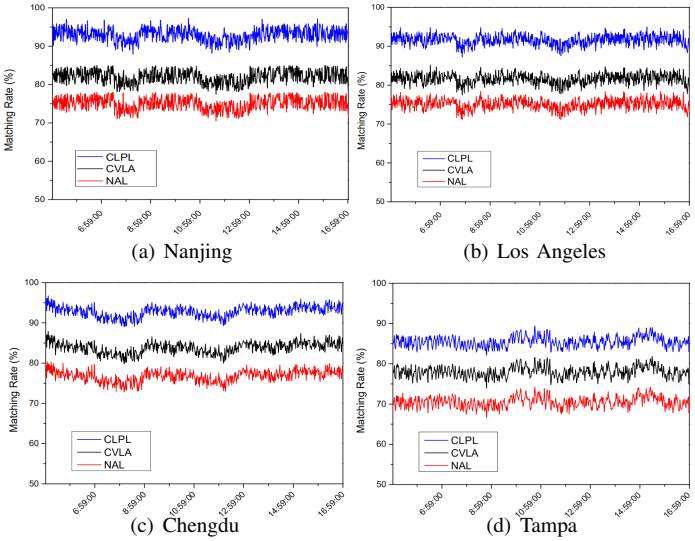
TABLE IV: The comparison of average matching rates among CLPL, CVLA and NAL under different ζ (%)(Nj/LA/Cd/Ta)

ζ (minutes)	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5
CLPL	80.9/75.5/85.1/68.9	85.0/79.8/88.2/71.2	88.5/83.2/89.47/75.5	92.2/86.3/93.1/80.1	93.2/90.4/95.9/82.5	94.9/92.1/96.8/84.0	92.5/89.8/94.1/85.3	91.0/87.3/93.5/77.8	92.1/85.1/91.3/74.7
CVLA	60.2/55.5/68.1/49.2	63.5/59.3/71.8/53.1	69.8/63.1/75.5/57.7	75.2/70.3/80.1/63.6	78.4/77.2/85.0/68.7	82.6/80.1/88.3/72.5	75.1/79.9/87.9/75.5	70.3/75.1/82.3/71.3	73.2/71.2/78.8/68.5
NAL	50.4/43.2/56.3/38.0	52.0/48.1/64.5/42.1	60.0/50.2/70.3/46.8	65.8/59.4/78.3/50.2	78.2/62.1/82.4/59.3	75.6/66.6/83.5/64.7	74.0/70.2/79.2/63.9	73.1/68.7/75.0/58.0	71.5/65.3/72.9/55.3



(a) Nanjing (b) Los Angeles (c) Chengdu (d) Tampa

Fig. 11: Matching rate of CLPL for different ζ values



(a) Nanjing (b) Los Angeles (c) Chengdu (d) Tampa

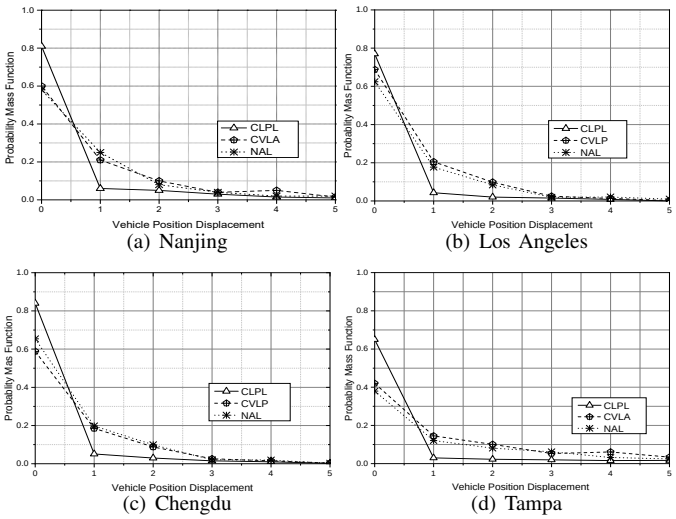
Fig. 12: Matching rate comparisons of CLPL, CVLA, and NAL with $\zeta = 3.0$

influences matching rate for different time-slot durations. For the 1.0-minute, CLPL with four hidden layers achieves the best performance, while CLPL with five layers has the best results for 2.0-minute, CLPL with six layers has the best results for both 3.0-minute and 4.0-minute. The difference of optimality with respect to the number of hidden layers could have been caused by the counteracting impacts of error reduction and the so-called *"overfitting"* phenomenon.

TABLE IV summarizes matching rates of the three algorithms with different time-slot durations. It can be seen that matching rates generally increase with ζ until around $\zeta = 3.0$ and then they decrease as ζ further increases. The increasing ζ introduces more history information of vehicle location in each time-slot, which leads to a more stable cluster structure and more accurate prediction result. However, the input data points for predictive localization decrease with the increasing ζ. If ζ exceeds a certain value, the prediction performance of CLPL deteriorates. In our experiments, $\zeta = 3.0$ is the turning point. Therefore, we will focus on $\zeta = 3.0$ henceforth unless specified otherwise.

Fig. 12(a) to (d) compares one-day-later prediction matching rates of the three algorithms with $\zeta = 3.0$ minutes as the time-slot duration, showing that both CVLA and NAL are inferior to CLPL. Other ζ values returned similar comparisons and have been omitted due to page limit.

Furthermore, the performance of CLPL is also evaluated in terms of vehicle location displacement and vehicle arrival time displacement. Vehicle location displacement refers to the number of road segments between the located road segment and the predicted road segment of a vehicle. As shown in Fig. 13 (a) to (d), the horizontal axis represents the vehicle location displacement while the vertical axis represents the



(a) Nanjing (b) Los Angeles (c) Chengdu (d) Tampa

Fig. 13: The vehicle location displacement analysis with $L_r = 30$ m

corresponding probability mass function. In this case, the probability mass function is observed under the length of the road segment $L_r = 30m$ (again, similar comparisons have been observed for $L_r = 50m$, $70m$, and $90m$, but omitted due to page limit). A clear advantage of the CLPL can be observed in Fig. 13, with much smaller probability masses other than zero displacement, i.e., accurate prediction.

In TABLE V, the average and the standard deviation of the probability mass function decrease with the increasing lengths of L_r; lower average values represent a lower vehicle location displacement while lower standard deviation values mean

TABLE V: The analysis of average and standard deviations under different L_r (Nj|LA|Cd|Ta)(average/standard deviation)

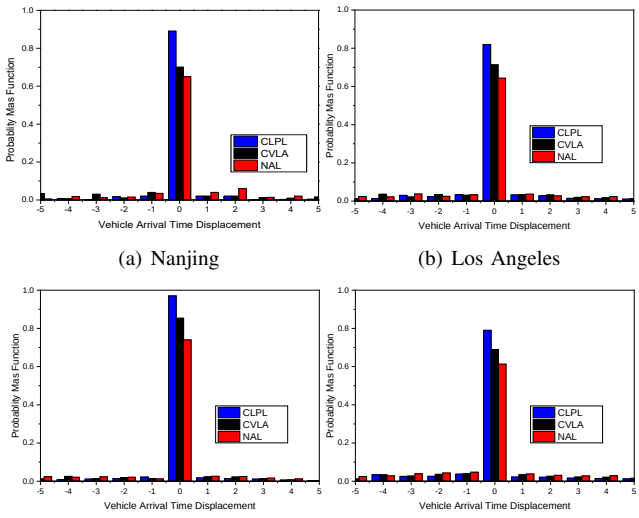
L_r (m)	30				50				70				90			
CLPL	0.90/0.95	0.86/0.91	1.10/1.25	0.68/0.78	0.75/0.87	0.72/0.85	1.02/1.13	0.62/0.75	0.66/0.81	0.59/0.78	0.95/1.03	0.53/0.66	0.52/0.72	0.49/0.67	0.87/0.98	0.42/0.58
CVLA	1.97/1.40	1.88/1.35	2.34/1.91	1.58/1.12	0.96/0.98	0.89/0.84	2.12/1.79	0.69/0.79	0.93/0.96	0.85/0.83	1.86/1.68	0.62/0.70	0.85/0.92	0.79/0.73	1.75/1.57	0.58/0.62
NAL	2.29/1.51	2.03/1.44	2.98/2.25	1.73/1.34	1.01/1.00	0.94/0.89	2.45/2.03	0.78/0.82	0.96/0.98	0.88/0.90	2.13/1.89	0.75/0.72	0.90/0.95	0.83/0.87	1.86/1.76	0.63/0.65

TABLE VI: Probability comparison with no vehicle arrival time displacement under different ζ (Nj/LA/Cd/Ta)

ζ (minutes)	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5
CLPL	0.798/0.618/0.852/0.681	0.854/0.541/0.882/0.708	0.890/0.759/0.930/0.753	0.917/0.813/0.961/0.776	0.923/0.823/0.972/0.791	0.924/0.837/0.976/0.795	0.927/0.840/0.978/0.799	0.929/0.846/0.981/0.804	0.932/0.851/0.984/0.811
CVLA	0.603/0.543/0.681/0.592	0.638/0.604/0.749/0.631	0.703/0.652/0.812/0.659	0.754/0.688/0.843/0.683	0.761/0.710/0.863/0.697	0.763/0.719/0.871/0.703	0.766/0.723/0.875/0.708	0.770/0.725/0.881/0.715	0.777/0.729/0.886/0.720
NAL	0.496/0.421/0.546/0.491	0.523/0.479/0.592/0.530	0.646/0.521/0.657/0.571	0.733/0.584/0.692/0.594	0.743/0.613/0.715/0.606	0.764/0.626/0.719/0.609	0.787/0.630/0.726/0.614	0.799/0.637/0.730/0.621	0.822/0.643/0.838/0.627

that the probability mass function distribution is relatively concentrated (*i.e.,* the vehicle location displacement is mainly around zero). CLPL has the least vehicle location displacement in all of the considered scenarios.

Since we are working on the prediction of the future vehicle location, we also need to take into account the possibility that vehicle may not arrive at the predicted road segment on time: early, delay, even never pass through the predicted road segment (*i.e.,* the arrival delay is considered as infinite). In such cases, the *vehicle arrival time displacement* is defined as the time-slot deviation between the arrival time and the predicted time when a vehicle reaches the road segment. Assuming that t_i^a represents the arrival time of the vehicle i while t_i^p is the predicted time of the vehicle i; the vehicle arrival time displacement is evaluated by $\lceil \overline{\Delta}/\Delta t \rceil$, which computes the number of deviated time-slots. Here, $\overline{\Delta} = t_i^a - t_i^p$.



(a) Nanjing (b) Los Angeles

(c) Chengdu (d) Tampa

Fig. 14: Vehicle arrival time displacement analysis with $\zeta = 3.0$

As shown in Fig. 14 (a) to (d), the horizontal axis is the vehicle arrival time displacement while the vertical axis is the corresponding probability mass function. Again, CLPL results are concentrated in zero arrival time displacement.

TABLE VI summarizes the probability mass function with zero vehicle arrival time displacement for the three algorithms, showing CLPL outperforming CVLA and NAL for about 30% and 60%, respectively.

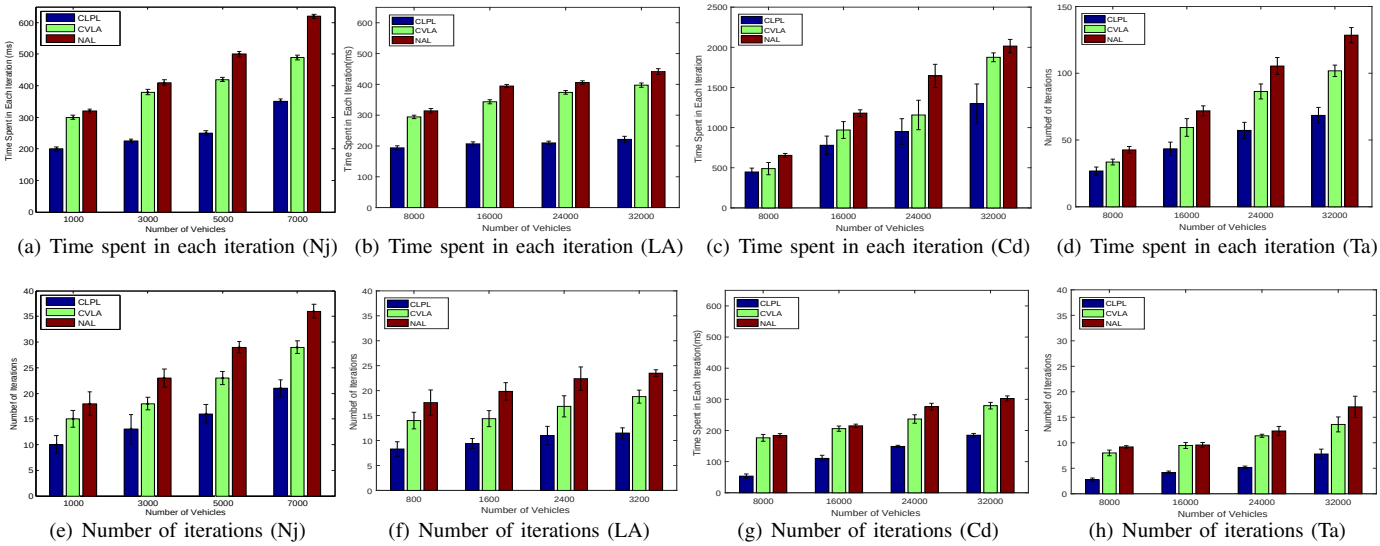
Fig. 15 shows a bar graph to compare the average time of each iteration and the number of iterations for these three schemes with increasing number of vehicles. These results are obtained by predicting locations after only one time-slot. The 95% confidence intervals are also shown with these bars. The result of multiplying them is the time spent for the prediction, which will increase with more time-slots included in the interval between the predicted time and the current time. It should be emphasized that the execution of CLPL can obtain the location of all vehicles at the same time instead of calculating separately for each vehicle. Computational cost generally increases with the number of vehicles, but CLPL is shown to have much lower costs.

VII. Conclusion

In order to obtain the location distribution information of vehicles in IoCV, we have analyzed the behavior correlation between vehicles. Then, a Behavior-based Clustering Method (BCM) has been designed to classify vehicles into clusters based on such a correlation. With BCM, we have further proposed a CLPL scheme to train on past data and provide long-term vehicle location distribution in IoCV. CLPL is able to obtain future vehicles' location distribution based on large amount of training data feeded into the multi-layer ISAE.

Extensive experiments have been conducted to evaluate BCM and CLPL. We have compared BCM with SPC and MPBC, mainly in terms of cluster formation delay and modularity Q. Although the BCM scheme has higher time and space cost, it is shown to be able to quickly stabilize with smaller number of cohesive clusters while starting with a large number vehicles. Vehicles ID, historical location, and the latest matrix are recorded using cloud storage. Considering the benefit of further storing data longer than one year to diminish rather quickly with time, the storage requirement is not excessively large in BCM. However, the balance of the storage requirement and prediction accuracy is not solved yet, and we leave this to our future work. The CLPL scheme has been compared with CVLA and NAL, mainly in terms of matching rates, vehicle location displacement, vehicle arrival time displacement, as well as computational time. We have shown that CLPL has a better matching rate, smaller vehicle location displacement, smaller vehicle arrival time displacement, and lower computational time, making it a great candidate for large-scale long-term vehicle predictive localization.

As a future direction, we plan to address the issue of dynamic adjustment on ζ and the number of hidden layers possibly using past data.



(a) Time spent in each iteration (Nj) (b) Time spent in each iteration (LA) (c) Time spent in each iteration (Cd) (d) Time spent in each iteration (Ta)

(e) Number of iterations (Nj) (f) Number of iterations (LA) (g) Number of iterations (Cd) (h) Number of iterations (Ta)

Fig. 15: Average time of each iteration and average number of iterations to predict locations after one time-slot for CLPL, CVLA, and NAL. 95% confidence intervals are also shown.

ACKNOWLEDGMENTS

This work was supported by the Liaoning Province Higher Education Innovative Talent Support Program; and the Fundamental Research Funds for the Central Universities under Grant No. DUT19JC22.

REFERENCES

[1] M. Chen, Y. Hao, H. Gharavi, and V. C. Leung, "Cognitive information measurements: A new perspective," *Information Sciences*, vol. 505, pp. 487–497, Dec. 2019.

[2] I. Lana, J. Del Ser, M. Velez, and E. I. Vlahogianni, "Road traffic forecasting: Recent advances and new challenges," *IEEE Intelligent Transportation Systems Magazine*, vol. 10, no. 2, pp. 93–109, Apr. 2018.

[3] W. Xu, H. Zhou, L. F. Cheng, N, W. Shi, J. Chen, and X. Shen, "Internet of vehicles in big data era," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 1, pp. 19–35, Dec. 2017.

[4] K. Jo, M. Lee, J. Kim, and M. Sunwoo, "Tracking and behavior reasoning of moving vehicles based on roadway geometry constraints," *IEEE Transactions on Intelligent Transportation Systems*, vol. PP, no. 99, pp. 1–17, Sep, 2016.

[5] S. A. Goli, B. H. Far, and A. O. Fapojuwo, "Cooperative multi-sensor multi-vehicle localization in vehicular adhoc networks," in *IEEE International Conference on Information Reuse & Integration*, San Francisco, CA, USA, Aug. 2015.

[6] E. I. Vlahogianni, M. G. Karlaftis, and J. C. Golias, "Short-term traffic forecasting: Where we are and where we're going," *Transportation Research Part C: Emerging Technologies*, vol. 43, pp. 3–19, Jun, 2014.

[7] S. Oh, Y. J. Byon, K. Jang, and H. Yeo, "Short-term travel-time prediction on highway: A review on model-based approach," *Ksce Journal of Civil Engineering*, no. 3, pp. 1–13, 2018.

[8] J. W. C. van Lint, "Reliable travel time prediction for freeways: bridging artificial neural networks and traffic flow theory," *Civil Engineering & Geosciences,TRAIL Research School*, Jan. 2004.

[9] F. Bonardi, S. Ainouz, R. Boutteau, Y. Dupuis, and P. Vasseur, "A novel global image description approach for long term vehicle localization," in *European Signal Processing Conference*, Kos, Greece, Aug. 2017.

[10] D. Jeong, M. Baek, and S. S. Lee, "Long-term prediction of vehicle trajectory based on a deep neural network," in *2017 International Conference on Information and Communication Technology Convergence (ICTC).IEEE*, Kos, Greece, Sep, 2017.

[11] D. Papakostas and D. Katsaros, "A rich-dictionary markov predictor for vehicular trajectory forecasting," in *IEEE 30th International Conference on Tools with Artificial Intelligence*, Volos, Greece, Dec, 2018.

[12] R. Madhavan and C. Schlenoff, "The effect of process models on short-term prediction of moving objects for unmanned ground vehicles," in *International IEEE Conference on Intelligent Transportation Systems*, Washington, WA, USA, Oct. 2004.

[13] X. Liu, "A short-term forecasting algorithm for network traffic based on chaos theory and svm," *Journal of Network & Systems Management*, vol. 19, no. 4, pp. 427–447, 2011.

[14] Z. Hou and X. Li, "Repeatability and similarity of freeway traffic flow and long-term prediction under big data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 6, pp. 1786–1796, Jan. 2016.

[15] F. Su, H. Dong, L. Jia, Y. Qin, and Z. Tian, "Long-term forecasting oriented to urban expressway traffic situation," *Advances in mechanical engineering*, vol. 8, no. 1, pp. 1–16, Jan. 2016.

[16] S. Qiao, H. Nan, J. Wang, R. H. Li, and X. Wu, "Predicting long-term trajectories of connected vehicles via the prefix-projection technique," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 7, pp. 2305–2315, Oct. 2017.

[17] H. Dia, "An object-oriented neural network approach to short-term traffic forecasting," *European Journal of Operational Research*, vol. 131, no. 2, pp. 253–261, Jun. 2001.

[18] W. Zheng and D. H. Lee, "Short-term freeway traffic flow prediction: Bayesian combined neural network approach," *Journal of Transportation Engineering*, vol. 132, no. 2, pp. 114–121, Feb. 2006.

[19] X. S. Lu, M. Zhou, L. Qi, and H. Liu, "Clustering-algorithm-based rare-event evolution analysis via social media data," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 2, pp. 301–310, Apr. 2019.

[20] J. Jokinen, T. Raty, and T. Lintonen, "Clustering structure analysis in time-series data with density-based clusterability measure," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 6, pp. 1332–1343, Nov, 2019.

[21] X. Xu, J. Li, M. Zhou, J. Xu, and J. Cao, "Accelerated two-stage particle swarm optimization for clustering not-well-separated data," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2018.

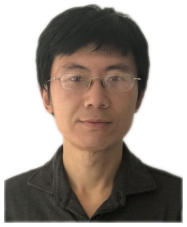
[22] J. Yang and J. Leskovec, "Overlapping community detection at scale: A nonnegative matrix factorization approach," in *Acm International Conference on Web Search & Data Mining*, Rome, Italy, Feb. 2013.

[23] Y. Zhang and D. Y. Yeung, "Overlapping community detection via bounded nonnegative matrix tri-factorization," in *the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, Beijing, China, Aug. 2012.

[24] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical Review E Statistical Nonlinear & Soft Matter Physics*, vol. 76, no. 2, p. 036106, Sep. 2007.

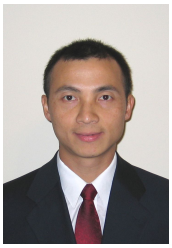
[25] S. Gregory, "Finding overlapping communities in networks by label propagation," *New Journal of Physics*, vol. 12, no. 10, pp. 2011–2024, 2009.

[26] L. Maglaras and D. Katsaros, "Social clustering of vehicles based on semi-markov processes," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 1, pp. 318–332, Jan. 2016.
[27] M. Ni, Z. Zhong, D. Zhao, M. Ni, Z. Zhong, and D. Zhao, "Mpbc: A mobility prediction-based clustering scheme for ad hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 9, pp. 4549–4559, Nov, 2011.
[28] N. Mattern, M. Obst, R. Schubert, and G. Wanielik, "Co-operative vehicle localization algorithm evaluation of the covel approach," Chemnitz, Germany, Mar. 2012.
[29] S. B. Cruz, T. E. Abrudan, Z. Xiao, N. Trigoni, and J. Barros, "Neighbor-aided localization in vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 10, pp. 2693 – 2702, Feb. 2017.
[30] D. Ravi, C. Wong, B. Lo, and G. Z. Yang, "A deep learning approach to on-node sensor data analytics for mobile or wearable devices," *IEEE J Biomed Health Inform*, vol. 21, no. 1, pp. 56–64, Jan. 2017.
[31] F. Wu, Z. Wang, Z. Zhang, Y. Yang, and Y. Zhuang, "Weakly semi-supervised deep learning for multi-label image annotation," *IEEE Transactions on Big Data*, vol. 1, no. 3, pp. 109–122, 2017.
[32] E. Principi, D. Rossetti, S. Squartini, and F. Piazza, "Unsupervised electric motor fault detection by using deep autoencoders," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 2, pp. 441–551, Mar. 2019.
[33] T. Mikolov, A. Deoras, D. Povey, L. Burget, and J. Černockỳ, "Strategies for training large scale neural network language models," in *IEEE Workshop on Automatic Speech Recognition & Understanding*, Waikoloa, HI, USA, Dec. 2011.
[34] M. Chen and Y. Hao, "Label-less learning for emotion cognition," *IEEE transactions on neural networks and learning systems*, Aug. 2019.
[35] J. Del Ser, E. Osaba, J. J. Sanchez-Medina, I. Fister, and I. Fister, "Bioinspired computational intelligence and transportation systems: A long road ahead," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 2, pp. 466–495, Feb. 2020.
[36] H. Xiong, B. Alipanahi, L. J. Lee, H. Bretschneider, D. Merico, R. K. Yuen, Y. Hua, S. Gueroussov, H. S. Najafabadi, T. R. Hughes *et al.*, "The human splicing code reveals new insights into the genetic determinants of disease," *Science*, vol. 347, no. 6218, pp. 124–144, Jan. 2015.
[37] F. Yang, J. Li, T. Lei, and S. Wang, "Architecture and key technologies for internet of vehicles: a survey," *Journal of Communications & Information Networks*, vol. 2, no. 2, pp. 1–17, Jun. 2017.
[38] S. Zhang, Y. Hu, and G. Bian, "Research on string similarity algorithm based on levenshtein distance," 2247-2251, Chongqing, China, Mar. 2017.
[39] M. Kasthuri, S. B. R. Kumar, and S. Khaddaj, "PLIS: Proposed language independent stemmer for information retrieval systems using dynamic programming," in *World Congress on Computing and Communication Technologies*, Tiruchirappalli, India, Feb. 2017.
[40] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E Statistical Nonlinear & Soft Matter Physics*, vol. 69, no. 2, p. 026113, Feb. 2004.
[41] R. B. Palm, "Prediction as a candidate for learning deep hierarchical models of data," *Technical University of Denmark*, vol. 5, 2012.
[42] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially private empirical risk minimization," *Journal of Machine Learning Research*, vol. 12, no. 29, pp. 1069–1109, 2011.
[43] I. Lana, J. J. Sanchez-Medina, E. I. Vlahogianni, and J. Del Ser, "From data to actions in intelligent transportation systems: a prescription of functional requirements for model actionability," *arXiv preprint arXiv:2002.02210*, Feb. 2020.
[44] L. Danon, A. Dazguilera, J. Duch, and A. Arenas, "Comparing community structure identification," *Journal of Statistical Mechanics*, vol. 2005, no. 09, pp. 219–228, 2005.



Kai Lin is an associate professor at the School of Computer Science and Technology, Dalian University of Technology. He received his M.S. and Ph.D. degrees in communication engineering from Northeastern University, China. His research interests include wireless communication, data mining and data fusion, big data analysis, mobile ad hoc networks, cyber physical systems, and sensor networks. He is an associate editor of recent advances in communications and networking technology and editor of several journals.

Lihui Li is a student at the School of Computer Science and Technology, Dalian University of Technology. He received his B.S. degree in internet of things engineering from Xidian University, China, in 2018. He is currently pursuing his M.S. degree, in computer software and theory from Dalian University of Technology. His current research interests are big data analysis, 5G, IoT, network spectrum allocation.



Jing Deng received the B.E. and M.E. degrees in electronics engineering from Tsinghua University, Beijing, China, in 1994 and 1997, respectively, and the Ph.D. degree from the School of Electrical and Computer Engineering, Cornell University, Ithaca, NY, USA, in 2002. He served as a Research Assistant Professor with the Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY, USA, from 2002 to 2004. He was with the Department of Computer Science, University of New Orleans, New Orleans, LA, USA, from 2004 to 2008. He visited the Department of Electrical Engineering, Princeton University, Princeton, NJ, USA, and the Wireless Information Network Laboratory, Department of Electrical and Computer Engineering, Rutgers University, Piscataway, NJ, USA, in 2005. He is currently a Professor with the Department of Computer Science, University of North Carolina at Greensboro, Greensboro, NC, USA. His current research interests include wireless network and security, information assurance, mobile ad hoc networks, and social networks. Dr. Deng received the Test-of-Time Award presented by the ACM Special Interest Group on Security, Audit and Control in 2013. He is an Associate Editor of the IEEE Transactions on Mobile Computing and served as an Editor for the IEEE Transactions on Vehicular Technology between 2008 and 2018.

Pasquale Pace (M'05) received his Ph.D. in information engineering in 2005 from the University of Calabria, Italy. He was a visiting researcher at the CCSR, Surrey, United Kingdom and at the Georgia Institute of Technology-USA. He is currently an Assistant Professor in telecommunications at the University of Calabria, Italy within the Department of Computer Engineering, Modeling, Electronics and Systems (DIMES). His research interests include cognitive and opportunistic networks, sensor and self-organized networks, interoperability of IoT platforms and devices. He is an Associate Editor of Telecommunication Systems Journal, and has also served several journals and Special Issues as Editor or Guest Editor. He has authored more than 90 papers in international journals, conferences and books.

Giancarlo Fortino (SM'12) is Full Professor of Computer Engineering at the Dept. of Informatics, Modeling, Electronics, and Systems of the University of Calabria (Unical), Italy. He received a PhD in Computer Engineering from Unical in 2000. He is also guest professor at Wuhan University of Technology (China), high-end expert at HUST (China), and senior research fellow at the ICAR-CNR Institute. He is the director of the SPEME lab at Unical as well as co-chair of Joint labs on IoT established between Unical and WUT and SMU Chinese universities, respectively. His research interests include agent-based computing, wireless (body) sensor networks, and IoT. He is author of 400+ papers in int'l journals, conferences and books. He is (founding) series editor of IEEE Press Book Series on Human-Machine Systems and EiC of Springer Internet of Things series and AE of many int'l journals such as IEEE TAC, IEEE THMS, IEEE IoTJ, IEEE SJ, IEEE SMCM, Information Fusion, JNCA, EAAI, etc. He is cofounder and CEO of SenSysCal S.r.l., a Unical spinoff focused on innovative IoT systems. Fortino is currently member of the IEEE SMCS BoG and of the IEEE Press BoG, and chair of the IEEE SMCS Italian Chapter.