AI-Enabled Spatial-Temporal Mobility Awareness Service Migration for Connected Vehicles

Chenglong Wang, *Student Member, IEEE*, Jun Peng, *Senior Member, IEEE*, Lin Cai, *Fellow, IEEE*, Hui Peng, Weirong Liu, *Member, IEEE*, Xin Gu, *Student Member, IEEE*, and Zhiwu Huang, *Member, IEEE*

***Abstract*—In the future 6G intelligent transportation system, the edge server will bring great convenience to the timely computing service for connected vehicles. To guarantee the quality of service, the time-critical services need to be migrated according to the future location of the vehicle. However, predicting vehicle mobility is challenging due to the time-varying of road traffic and the complex mobility patterns of vehicles. To address this issue, a spatial-temporal awareness proactive service migration strategy is proposed in this paper. First, a spatial-temporal neural network is designed to obtain accurate mobility by using gated recurrent units and graph convolutional layers extracting features from spatial road traffic and multi-time scales driving data. Then a proactive migration method is proposed to guarantee the reliability of services and reduce energy consumption. Considering the reliability of services and the real-time workload of servers, the migration problem is modeled as a multi-objective optimization problem, and the Lyapunov optimization method is utilized to obtain utility-optimal migration decisions. Extensive simulations based on real-world datasets are performed to validate the performance of the proposed method. The results show that the proposed method achieved 6% higher prediction accuracy, 10% lower dropping rate, and 10% lower energy consumption compared to state-of-the-art methods.**

***Index Terms*—Lyapunov optimization, proactive service migration, spatial-temporal mobility prediction, vehicular edge networks.**

I. Introduction

IN RECENT years, the rise of artificial intelligence (AI) has pervaded various areas from the 6G-enabled edge computing networks to the future intelligent transportation systems. With the help of AI and 6G, it will be possible for intelligent transportation systems to achieve advanced services for connected vehicles such as driving assistance and autonomous driving [1]. In the 6G era, connected vehicles expect seamless proximal services, which require a tremendous amount of real-time data processing, hyper-fast data rate, and extremely low latency [2]. However, the high-speed movement of vehicles makes it difficult to guarantee the ultimate experience of the aforementioned time-critical and low fault tolerance services. It gives rise to a high possibility of impairment of quality of service (QoS) and latency increasing [3]. This will further lead to service level objective violations, service interruptions, and even serious traffic accidents [4]. Therefore, it is critical to develop a mechanism to migrate services according to the movement of vehicles.

The problem of developing a robust service migration mechanism for the intelligent transportation system is challenging [5], [6]. The pivotal issue of service migration is to guarantee that the services can be migrated to the optimal edge server according to the movement of vehicles [7]. However, the complex mobility patterns of vehicles and time-varying road traffic make it challenging to accurately predict the mobility of vehicles. To avoid significant adverse effects such as service interruption caused by misprediction, the prediction models need to be extremely accurate. Another key issue is to reduce migration costs while ensuring QoS requirements [8]. During service migration, the edge server needs to consume energy to transmit service data to another edge server [9]. Frequent service migration due to vehicle movement will cause a large amount of data transmission, resulting in unacceptable energy consumption [10]. On the contrary, if the service migration is not based on vehicle movement, the proximity computing requirement of vehicle services cannot be guaranteed. This makes it hard to make an optimal migration decision to balance energy consumption and QoS requirement.

Over the past few years, service migration has been explored extensively [11], [12], [13], [14]. Many existing migration methods focus on migrating services *reactively* [15], which migrate services based on specific thresholds for various metrics such as quality loss tolerance [16], proximity requirements [17], and energy consumption [18]. However, reactive service migration can not foresee the workload of the server, which will impair the QoS. When the target server is working under a high workload, the server will block the migration requests to avoid overload, which will lead to service interruption. To avoid this problem, the *proactive* way is increasingly proposed.

Proactive service migration is a predictive approach that migrates services before the considered metrics reach their thresholds [19]. The performance of the proactive method largely depends on the prediction accuracy of vehicle mobility [20].

Manuscript received 12 December 2022; revised 17 April 2023; accepted 25 April 2023. Date of publication 1 May 2023; date of current version 6 March 2024. This work was supported by the National Natural Science Foundation of China under Grants 62172448 and 62003358. Recommended for acceptance by H. Wu. *(Corresponding author: Hui Peng.)*

Chenglong Wang, Jun Peng, Hui Peng, and Weirong Liu are with the School of Computer Science and Engineering, Central South University, Changsha 410083, China (e-mail: chenglongwang@csu.edu.cn; pengj@csu.edu.cn; hui-peng@csu.edu.cn; frat@csu.edu.cn).

Lin Cai is with the Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC V8W 3P6, Canada (e-mail: cai@ece.uvic.ca).

Xin Gu and Zhiwu Huang are with the School of Automation, Central South University, Changsha 410083, China (e-mail: guxin15@csu.edu.cn; hzw@csu.edu.cn).

This article has supplementary downloadable material available at https://doi.org/10.1109/TMC.2023.3271655, provided by the authors.

Digital Object Identifier 10.1109/TMC.2023.3271655

Recently, in order to predict vehicle mobility accurately, works in [21], [22], [23], [24] proposed several mobility prediction methods based on probabilistic models [25], the Markov chain [26], and neural networks [27]. However, considering the migration failure caused by misprediction and real-time server workload, how to optimize the migration decision is still an underexplored problem. Furthermore, combining the dynamically changed traffic conditions to improve mobility prediction is a promising open issue.

To make appropriate migration decisions, it is crucial to accurately predict the mobility of vehicles in combination with dynamic traffic conditions. In this paper, gated recurrent units and graph convolutional networks are used to extract temporal trends and cross-correlated road features from real-time driving data, history trajectory data, and road traffic data. The real-time driving data and the historical trajectory from other vehicles are formulated as a time sequence processed by two homogeneous gated recurrent units. The road traffic is modeled as a graph structure processed by a two-layer graph convolutional network. By fusing spatial and temporal features, accurate mobility prediction can be obtained. Meanwhile, considering the migration failure caused by misprediction and the real-time server workload, the migration problem is modeled as a multi-objective optimization problem. Based on the prediction results, a mobility awareness proactive migration method for vehicular edge computing networks is proposed to balance the QoS of vehicles and the energy consumption of the servers.

In light of these discussions, the main contributions of this paper are summarized as follows:

- An AI-enabled intelligent vehicle mobility prediction network is designed for edge computing service in intelligent transportation systems. The designed prediction network can predict vehicle mobility by capturing the complicated spatial-temporal traffic dependencies from historical traffic data and real-time driving data. Incorporating gated recurrent units and graph convolutional networks, the designed prediction network can achieve higher mobility prediction accuracy compared to other state-of-the-art prediction methods.
- To guarantee the reliability of intelligent edge services for connected vehicles, a proactive migration mechanism is devised. Considering the reliability of services and the real-time workload of servers, the migration problem is modeled as a multi-objective optimization problem. Solving by Lyapunov optimization, the proposed method can make utility-optimal migration decisions to best balance the quality of service and migration cost.
- We perform extensive simulations on the realistic vehicle trajectories in Bologna and Luxembourg to validate the superior performance of the proposed proactive migration method against the state-of-the-art methods. The results show that the proposed method performs best in terms of prediction accuracy, service dropping rate, energy cost, and the queue length of servers. The proposed method achieved up to 6% higher prediction accuracy, 10% lower dropping rate, and 10% lower energy consumption compared to state-of-the-art methods.

The rest of this paper is organized as follows. Relative work is introduced in the next Section. The system model is presented in Section III. Section IV introduced the spatio-temporal mobility prediction network. The formulation of the problem is presented in Section V. Whereas the numerical results based on the realistic vehicle trajectory in Bologna are presented in Section VI, followed by the concluding remarks and future work in Section VII.

II. Related Works

Service migration is a key mechanism to maintain the continuity of service when a vehicle changes the coverage of base stations [28]. A well-designed migration mechanism can avoid energy consumption and QoS impairment [29]. Generally, service migration methods can be divided into two categories: reactive migration and proactive migration.

The reactive method migrates services based on specific thresholds for various metrics such as quality loss tolerance, proximity requirements, and energy consumption. To ensure QoS during reactive migration, a distributed traffic steering live migration framework was proposed in [11], which can maintain QoS through dynamic path planning to avoid service interruption. In [12], a delay-aware bandwidth slicing migration method was designed to meet the different delay requirements of migration traffic and non-migration traffic by using the bandwidth slicing mechanism. Considering the trade-off between migration cost and user-perceived quality, a migration method based on the Markov decision process was proposed in [13] for cloud network migration decisions. Taking the service placement into consideration, the work in [14] further applied the same framework to the edge. By solving the performance optimization problem under long-term services quality constraints, a Follow-Me edge service migration method was developed.

A drawback of these reactive methods is that migration is triggered by a certain threshold of considered metrics. Since reactive methods cannot predict the mobility of vehicles, they assumed that all migration requests will be accepted by the target server. However, a server with a high workload may block some migration requests to guarantee the quality of service in progress, which will cause migration failure. Although it can be alleviated by the handover or migration again to other idle servers, considerable energy consumption is unacceptable.

Recently, some proactive service migration methods considering vehicle mobility have been investigated. Aiming to minimize the total energy consumption, an energy-efficient service migration method was proposed in [21], which can make proactive migration decisions according to the vehicle trajectories. By modeling the trajectories as a Markov chain, the future locations of the user can be obtained, and the contents can be migrated proactively.

Some works further incorporated machine learning methods into proactive migration. In [22], a LSTM-based user mobility prediction was proposed for service management, which can achieve the best match between the expected service quality of users and the optimal utilization of MEC resources. In [23], a joint migration and mobility optimization approach

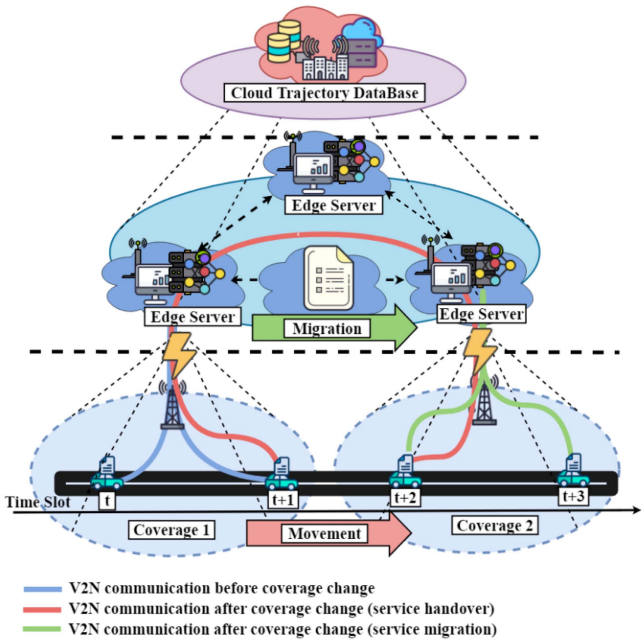


Fig. 1. The system structure of the vehicular edge computing network. There are three types of V2N communication links in the system. The vehicle accesses the edge server and gets feedback through the link shown as the blue line. After coverage changes, the vehicle accesses the server and gets feedback by service handover through the backhaul link shown as the red line. After the services are migrated to the accessed server, the vehicle access the server through the link shown as the green line.

was designed to minimize the migration cost. Through mobility prediction and path planning, the formulated optimization problem can be transformed into a composite utility function and solved by deep Q-learning networks. Moreover, a mobility prediction framework incorporating gated recurrent units and a convolutional neural network was proposed in [24]. It can predict the future vehicle location according to the real-time azimuth of the vehicle. Based on the prediction results, a proactive migration decision is made to reduce energy consumption.

However, the existing work ignored the migration failure caused by misprediction. Misprediction will cause the service to be migrated to the wrong server, which will further lead to high energy consumption and even service interruption. On the other hand, the mobility prediction in the existing work is not accurate enough. The time-varying road travel time and dynamically changed traffic conditions were not considered, which are very helpful to improve prediction accuracy.

III. System Model

A. System Overview

The vehicular edge computing network is shown in Fig. 1. Each base station (BS) provides wireless communication services for vehicles in its coverage, and an edge server is deployed in each BS for computing services. The coverage of the BSs is non-overlapping and vehicles can request services from an edge server through the BS. The edge server that hosts the vehicle's requests is denoted as the source server s_{host}.

Due to the high wire-deployment costs and ultra-dense BS deployment mentioned in [30], we assume that edge servers are interconnected via a wireless backhaul network. If a vehicle leaves the coverage of s_{host}, the service migration or handover will be performed to guarantee the service continuity of this vehicle through the backhaul network.

An example of the handover and migration process is shown in Fig. 1. Initially, vehicles access the source server s_{host} through the access BS directly as shown in the blue line. When vehicles leave the coverage of s_{host}, all the data to and from these vehicles are handover from their source server s_{host} through the backhaul link to keep service continuity shown in the red line. If the backhaul link cannot meet the QoS requirement, the services of those vehicles will be migrated to the edge server in the current cell (named access server) through the green arrow.

The migration process consists of two steps. First, the access server will copy common application data locally to support the same services. Then the private data of migrated vehicle will transmit from the source server s_{host} to the access server. The handover mechanism will ensure service continuity during migration. Once the migration has been done, the access server will become the new source server, and the vehicle can request service from it through the connection shown in the green line.

To design a suitable migration method, two main issues are addressed in this paper : (i) How to accurately estimate the mobility of vehicles and workloads of edge servers; (ii) Whether to migrate them considering the energy cost and the QoS requirement of vehicles. To address these issues, the detailed model formulation is given in the following subsections, and the main notations used in this paper are listed in Table I.

B. Energy Cost

As a critical indicator, the migration decision is highly influenced by energy consumption. First, we will formulate the energy consumption of three types of vehicles: directly connected vehicles, handover vehicles, and migration vehicles.

In this paper, time is divided into time slots t. Suppose the base station (BS) and the edge server are collocated, and there are m edge servers in the certain area defined as a set $s_i \in S\{s_1, s_2, \cdots s_m\}$. The vehicles served by edge server s_i at time t are denoted as $v \in V_i(t)$.

Each edge server serves three types of vehicles, divided into the directed connected set $V_i^D(t)$, the handover set $V_i^H(t)$, and the migration set $V_i^M(t)$. Vehicles connected to the BS directly in the same cell as that of server s_{host} belong to $V_i^D(t)$. Those connected to different cells from that of server s_{host} through the handover mechanism belong to $V_i^H(t)$. Those connected to different cells from that of server s_{host} during service migration belong to $V_i^M(t)$.

In time slot t, the relation between $V_i(t)$ and those three sets can be defined as

$$V_i^D(t) + V_i^H(t) + V_i^M(t) = V_i(t). \quad (1)$$

TABLE I
SYMBOL TABLE

Symbol	Description
S	The set of edge server
V	The set of vehicle
V^D	The set of direct access vehicle
V^H	The set of service handover vehicle
V^M	The set of service migration vehicle
M_v	The migration decision of vehicle
δ	The migration indicator on whether migrate service
φ	The migration indicator on where to migrate service
P_s	The transit power of BS
P_v	The transit power of vehicle
P_b	The transit power of BS in backhaul network
R	The transmission rate between the vehicle and the access BS
R_b	The backhaul transmission rate
c_{up}	The packet size of upload link traffic
c_{down}	The packet size of download link traffic
e_d	The transmission energy cost for direct access vehicle
e_h	The transmission energy cost for service handover vehicle
e_m	The transmission energy cost for service migration vehicle
e_{copy}	The self-copy energy cost for migrating service
$\mu_{(j,i)}$	The number of vehicles from server s_j to server s_i
ϕ_i	The number of services served by server s_i
α_i	The vehicle arrival rate of server s_i
β_i	The vehicle departure rate of server s_i
γ_i	The service arrival rate of server s_i
Q_i	The queue length of server s_i
C	The capacity of server
C_L	The resource requirement of one service request
p_g	The probability of vehicle request a service
p_{on}	The probability of the service in the on-state
p_d	The probability of new service is dropped
N_i	The served vehicle number of server s_i
n_m	The blocked number of migration services
n_n	The blocked number of new services
χ_{bm}	The utility loss for blocking a migration service
χ_{bn}	The utility loss for blocking a new service

When vehicle v leaves the coverage of the source server, the service migration or handover will be performed to guarantee the service continuity.

The $M_v(t) = (\delta(t), \varphi(t))$ denotes the decision on whether and where to migrate the services. Where $\delta(t) \in \{0, 1\}$ denotes whether to migrate services, and $\varphi(t) \in \{1, 2, \cdots m\}$ indicates where to migrate.

If $\delta(t) = 0$, the services will not be migrated. Vehicle v connecting to the new access BS will be transferred from direct connection set V_i^D to handover set V^H.

If $\delta(t) = 1$, the services of v will be migrated to $s_{\varphi(t)}$, and v will be transferred to the migration set V_i^M until the migration process is finished.

Let c_{up} and c_{down} be the amount of upload and download requests in time slot t respectively, the direct connection transmission cost e_d can be given as

$$e_d = \frac{c_{up} * P_v + c_{down} * P_s}{R(t)}, \tag{2}$$

where P_s and P_v are the transmission power of BS and the transmission power of the vehicle. $R(t)$ is the transmission rate between v and the access BS, which can be given as

$$R(t) = W \cdot \log_2 \left(1 + \frac{P_s \cdot d(t)_{v,s}^{-\sigma}}{I}\right), \tag{3}$$

where $d(t)$ is the distance between the source server and the vehicle v. σ is the path loss exponent, and I is the random Gaussian noise.

In the case of handover, the requests of vehicle v need to be handover from the accessed BS to the source server, so the additional backhaul cost $\frac{(c_{up}+c_{down})*P_b}{R_b}$ will be added to the energy cost. Where P_b and R_b are the backhaul transmission power and rate. The handover energy cost e_h can be represented as

$$e_h = e_d + \frac{(c_{up} + c_{down}) * P_b}{R_b}. \tag{4}$$

In the case of migration, the cost consists of the transmission part and the self-copy part. The transmission part is to migrate the private data of vehicles, and the self-copy part is to copy common application data locally to support the same type of services in a new edge server. Accordingly, the migration cost e_m can be given as

$$e_m = e_d + \frac{(c_{up} + c_{down} + c_s) * P_b}{R_b} + e_{copy}. \tag{5}$$

where c_s is the size of vehicle's private data, and e_{copy} is the self-copy energy cost, which can be treated as a constant.

Based on (2), (4), and (5), if the server being migrated to can serve the vehicle in the following time, the migration energy cost $e_i(t)$ of can be formulated as

$$e_i(t) = \begin{cases} (1 - \delta(t))e_h + \delta(t)e_m & if \quad v_i \notin V_i^D, \\ e_d & otherwise. \end{cases} \tag{6}$$

Due to inaccurate prediction, there is the risk of migrating service to the wrong server, so it is necessary to account for the energy cost caused by prediction errors. Migration errors are categorized into three types: premature migration, late migration, and wrong migration.

The premature migration denotes the situation that the migration is performed too early. In other words, the predicted server will be a future accessed edge server. In this case, the packet from the server needs to be transmitted back to the current access server by the handover mechanism. Thus, the additional handover cost e_h will be added.

The late migration denotes the situation that the migration is performed too late. In other words, the predicted server is the accessed server in time slot t, but the vehicle will change the accessed server in later slot. In this case, vehicles will generate new private data, which needs to be retransmitted in the future. Thus the additional transmission energy cost $(c_s * P_s(t))/R$ will be added.

In cases of wrong migration, the predicted server is neither the next edge server in the future nor the previous edge server in the past. The predicted location is wrong completely. Thus the whole migration process is wasted, and the migration cost e_m will be added.

In summary, the energy cost can be rewritten as

$$E(t) = \begin{cases} e_i(t) + \frac{c_s * P_s(t)}{R} & if \quad \text{late migrate} \\ e_i(t) + e_h & if \quad \text{premature migrate} \\ e_i(t) + e_m & if \quad \text{wrong migrate} \\ e_i(t) & otherwise \end{cases} \tag{7}$$

C. Resource Reservation

Resource reservation is a crucial mechanism to guarantee the QoS. Any migration decision $M(t)$ needs to consider the target edge server's load. Services could be migrated if and only if the target edge server has the computational resource to host those services. The QoS requirement on queue length and the drop rate is formulated in this section.

Assume that the future locations of vehicles can be obtained by the mobility prediction. Based on the prediction, the new vehicle arrival rate $\alpha_i(t+1)$ of server s_i in time slot $t+1$ can be defined as

$$\alpha_i(t+1) = \sum_{j=1;j\neq i}^{m} \mu_{(j,i)}(t+1) \cdot \delta(t), \tag{8}$$

where $\mu_{(j,i)}(t+1)$ is the number of vehicles from server s_j to server s_i, which can be obtained by the mobility prediction. Similarly, the departure rate $\beta_i(t+1)$ given by

$$\beta_i(t+1) = \sum_{j=1;j\neq i}^{m} \mu_{(i,j)}(t+1) \cdot (1-\delta(t)). \tag{9}$$

Assuming that each vehicle has the probability p_g to generate a new service request, based on (8) and (9), the service arrival number of server s_i in time slot $t+1$ can be given as

$$\gamma_i(t+1) = [\max\{\phi_i(t) - \beta_i(t+1), 0\} + \alpha_i(t+1)] \cdot p_g, \tag{10}$$

where $\phi_i(t)$ is the number of vehicles served by server s_i in t.

Assume that all the edge servers have the same capacity C_L. Let C be the resource requirement of one service request. Based on the service arrival rate $\gamma_i(t+1)$, the queue length of s_i can be given as

$$Q_i(t+1) = \max\left\{Q_i(t) - \frac{C_L}{C}; 0\right\} + \gamma_i(t+1). \tag{11}$$

Considering time-division multiplexing, an On-Off service queue model [4] is used to formulate the usage of the server.

For ongoing service, there are two states to indicate whether there are service requests generated. On means the customer generates service requests in the time slot and Off otherwise.

Let a and b be the average duration of busy and idle, and N denotes the maximum served vehicular customer number of the edge server. The probability of the service in the "on" state can be represented as

$$p_{on} = \frac{a}{a+b}. \tag{12}$$

Since the ongoing services don't always generate service requests, the max served customer number $N_{\max}$ can be larger than $\frac{C_L}{C}$ to serve more customers by defining a threshold of service drop rate related to the QoS requirement.

Theorem 1 (The On-off resources model of max served customer number):

Let k be the threshold of service drop rate, the relation between the max served customer number $N_{\max}$ and k can be represented as

$$N_{\max} = \frac{C_L}{p_{on}} - \frac{1}{p_{on}}\left[\sqrt{4\ell(C_L+\ell)} - 2\ell\right],$$
$$\text{where } \ell \approx k^2(1-p_{on})/4. \tag{13}$$

Proof can be seen in Appendix-A, available in the online supplemental material.

If served customer number $N > N_{\max}$, the server will work on the overload state. The services arriving beyond the max served customer number $N_{\max}$ would be dropped. Suppose that there are N ($N > N_{\max}$) customers on the On-state, the service drop rate can be represented as

$$p_d = \sum_{i=C_L}^{N} \binom{N_{\max}}{i} {p_{on}}^i (1-p_{on})^{i-N_{\max}}. \tag{14}$$

A high drop rate will severely affect user experience. It is necessary to reserve resources to avoid service dropping for ongoing services.

If $N > N_{\max}$, either some migrated services or new services are blocked. In the case of blocking migration, the whole migration process is wasted, and the blocking utility loss $E_b(t)$ can be represented as

$$E_b(t) = n_m \cdot \chi_{bm} + n_n \cdot \chi_{bn}, \tag{15}$$

where χ_{bm} and χ_{bn} denote the new service blocking loss and the migration service blocking loss. n_m and n_n are the blocked number of migration services and new services. Since a blocked migration service can still be served by its previous server through the handover mechanism, the χ_{bn} is higher than χ_{bm}.

D. Problem Formulation

We solve the problem in two steps. First, the mobility prediction of all vehicles is obtained to calculate the vehicle arrival rate $\alpha_i(t+1)$ and the vehicle departure rate $\beta_i(t+1)$. To address this issue, a spatial-temporal neural network will be introduced in Section IV.

Second, an optimal migration decision $M(t) = (\delta(t), \varphi(t))$ is devised based on the mobility prediction results. An utility function $G(M(t))$ is defined to make optimal migration decisions considering the energy cost, the queue length of server, and service drop rate.

Based on (15), the utility function $G(M(t))$ can be represented as

$$G(M(t)) = w_1 N\chi - w_2 E_b(t) - w_3 E(t), \tag{16}$$

where the χ denotes revenue for serving a customer, and the w_1, w_2, and w_3 are the weight to normalize different quantity.

The objective of this paper is to maximize the utility function by selecting the optimal migration decisions $M(t)$, which can be formulated as follows

$$\max_{M} \sum_{s_i \in S} \sum_{v \in s_i} G(M(t)),$$
$$\text{s.t. (a) } \delta(t) \in \{0,1\}, \forall t,$$
$$\text{(b) } \varphi(t) \in \{1,2,\cdots m\}, \forall t. \tag{17}$$

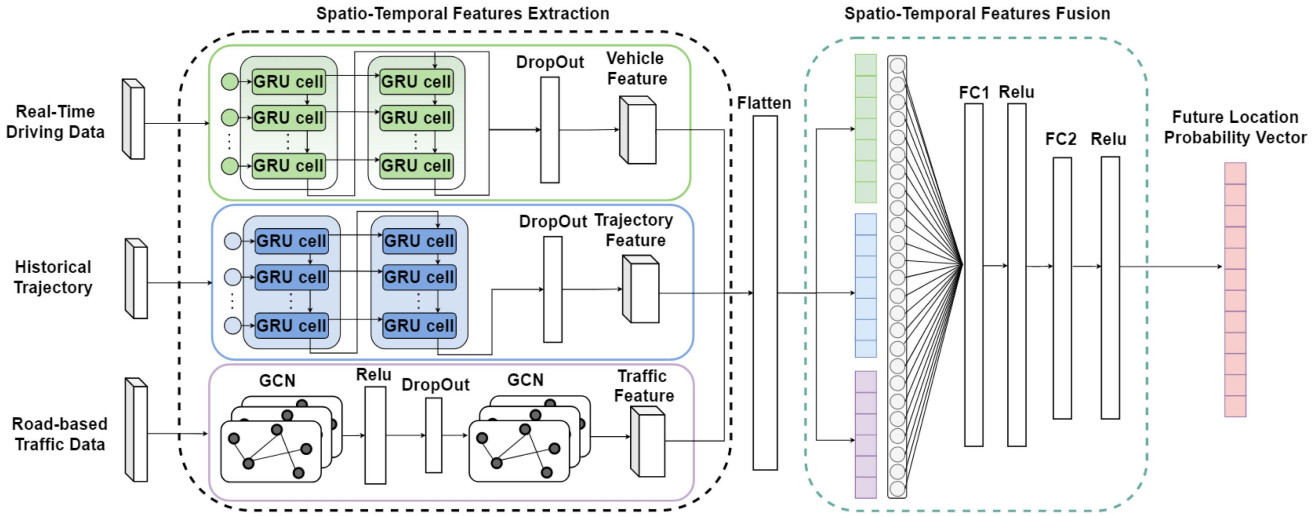


Fig. 2. The proposed AI-enabled spatio-temporal mobility prediction framework to obtain the future location of connected vehicles. Real-time drive data and historical trajectory are handled by two gated recurrent unit networks to extract temporal features, and road traffic data are handled by graph convolutional networks to extract spatial features.

To solve this problem, a Lyapunov optimization based service proactive migration method will be presented in Section V.

IV. Spatio-Temporal Mobility Prediction

To migrate services proactively, the first and most important step is to obtain an accurate vehicle mobility prediction. In this section, a spatio-temporal mobility prediction network is designed to improve the accuracy of prediction.

As shown in Fig. 2, three types of data are collected for mobility prediction. The driving data and the historical trajectory data are collected through speed, azimuth, acceleration, and coordination of vehicle location, which depict the mobility characteristics of vehicles. The road-based traffic data are collected from the road sections through the mean travel time, mean speed, max speed, vehicle count, and the percentage of lane occupancy.

To fully capture the potential correlations of the data, the proposed network uses three neural networks to extract features. The real-time drive data and the historical trajectory from other vehicles are formulated as a time sequence handled by two homogeneous 2-layer gated recurrent units (GRU). The road-based traffic data is modeled as a graph structure handled by a 2-layer graph convolutional network (GCN). Then three extracted features are flattened into a one-dimensional vector as the input of the 2-layer fully connected network.

The reason why we use the spatio-temporal neural structure is that the future locations of certain vehicles depend not only on their underlying characteristics (such as speed and acceleration) but also on the road traffic and the previous trajectories of other vehicles [31], [32]. As shown in Fig. 3(a), vehicles passing through the same trajectories departure in two recent time slots may have the same trend in travel speed. Since the traffic condition and road congestion in a certain period are similar, considering the trajectories of other vehicles is very useful to improve the accuracy of prediction.

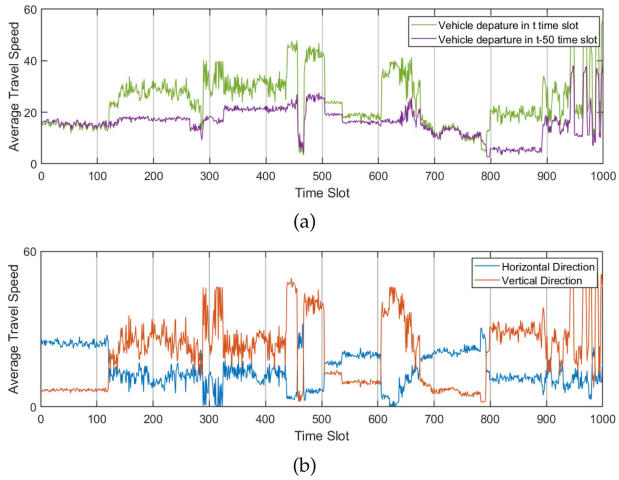
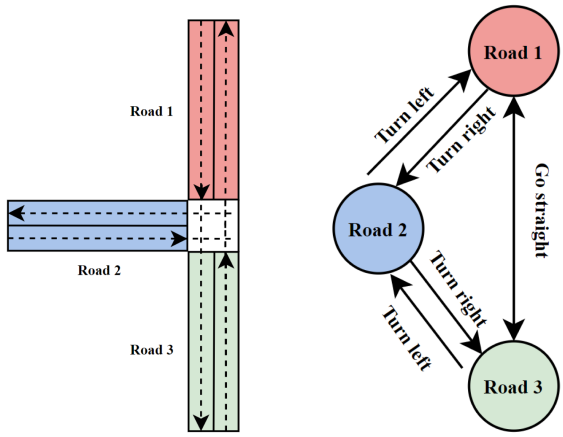


Fig. 3. The illustration of potential relevance on historical trajectory and road traffic. (a) The potential spatial relevance between vehicles on the same trajectories at different times. (b) The potential temporal relevance between traffic flow in the horizontal and vertical directions at the same time slot.

Besides, the traffic condition of adjacent roads is also correlated. As shown at the intersection of the two roads in Fig. 3(b), there are two opposing traffic speed trends in horizontal and vertical directions. If the horizontal direction is open to use, the vertical direction is blocked. Moreover, the next road section's travel speed is related to the previous road section's travel speed. If the previous road is congested, the next roads are more likely to be congested in the future. Therefore, road traffic data can also be used to improve prediction accuracy.

A. Gated Recurrent Unit for Time Sequences

Recurrent neural network (RNN) is a suitable tool to extract hidden temporal inter-correlation from sequential data. It has been proved in [33], [34] that among all different types of RNN,



(a) Real road topology (b) Graph structure

Fig. 4. The illustration of converting real-world road topology to the graph structure. (a) The real-world road topology includes three road sections. (b) The corresponding graph structure.

GRU has fewer parameters, faster convergence speed, and has almost the same performance as LSTM. To avoid over-fitting and excessive parameters, GRU is chosen as a feature extractor for the temporal data in this paper.

For GRU, the reset gate $\boldsymbol{r}$ and the update gate $\boldsymbol{z}$ are the core components to capture long-term dependencies. The reset gate is used for incorporating input and previous memory, and the update gate $\boldsymbol{z}$ is used to control the preservation of previous memory. The transition functions in hidden units of GRU are given as follows:

$$\begin{aligned}
\boldsymbol{r}_t &= \sigma\left(\boldsymbol{W}_{ir}\boldsymbol{x}_t + \boldsymbol{b}_{ir} + \boldsymbol{W}_{hr}\boldsymbol{h}_{(t-1)} + \boldsymbol{b}_{hr}\right), \\
\boldsymbol{z}_t &= \sigma\left(\boldsymbol{W}_{iz}\boldsymbol{x}_t + \boldsymbol{b}_{iz} + \boldsymbol{W}_{hz}\boldsymbol{h}_{(t-1)} + \boldsymbol{b}_{hz}\right), \\
\boldsymbol{n}_t &= \tanh\left(\boldsymbol{W}_{in}\boldsymbol{x}_t + \boldsymbol{b}_{in} + \boldsymbol{r}_t * \left(\boldsymbol{W}_{hn}\boldsymbol{h}_{(t-1)} + \boldsymbol{b}_{hn}\right)\right), \\
\boldsymbol{h}_t &= (1 - \boldsymbol{z}_t) * \boldsymbol{n}_t + \boldsymbol{z}_t * \boldsymbol{h}_{(t-1)},
\end{aligned} \tag{18}$$

where $\boldsymbol{W}$ and $\boldsymbol{b}$ are the model parameters for training. $\boldsymbol{h}_t$ is the hidden state at time t, $\boldsymbol{x}_t$ is the driving data at time t, $\boldsymbol{h}_{t-1}$ is the hidden state of the layer at time $t-1$, and $\boldsymbol{r}_t$, $\boldsymbol{z}_t$, $\boldsymbol{n}_t$ are the reset, update, and new gates, respectively. $\sigma(\cdot)$ is the sigmoid function.

Two GRU networks are used to handle driving data $\boldsymbol{X}_t^{drive}$ and historical trajectory $\boldsymbol{X}_t^{trajectory}$. For real-time driving data, each layer has 8 GRU cells, and the dropout probability is set to 0.3. Each layer's final state $\boldsymbol{h}_n$ will be the output of the two-layers GRU part. For historical trajectory, the GRU cell number is 8, and the output will be the final state $\boldsymbol{h}_n$ of the second layer to capture deep features on the whole trajectory.

B. Graph Convolutional Network for Road Traffic

To capture the complex spatial dependence features in road-based traffic data, the first step is to convert road traffic into a graph structure. An example of converting real road topology to graph structure is given in Fig. 4.

In Fig. 4(a), the real road topology consists of three road sections. First, road sections are converted to vertices shown in Fig. 4(b). According to the road connectivity, edges are added to this graph. For example, when vehicles turn right from road 1, they will reach road 2, and there is an edge connecting road 1 to road 2. Similarly, vehicles that go straight will reach road 3, so there is an edge from road 1 to road 3. In this way, the whole trajectory and all roads adjacent to it can be modeled as a graph G.

In this paper, we aim to learn the future traffic trend from adjacent road sections. So based on the location of the vehicle, graph G contains the current road section and 4 adjacent sections. In graph G, each vertex contains 4 road features including the mean travel time, mean speed, max speed, and the percentage of lane occupancy.

Based on the formed graph, a graph convolutional network is designed to capture the complex spatial dependence. As shown in the purple part of Fig. 2, the graph convolutional network (GCN) contains two GCN layers. The channel number of the first GCN layer is 25, and the output channel number is 5. The input channel number of the second GCN layer is 5, and the output channel number is 1. Similar to the GRU layer, a dropout probability is set as 0.3. The graph convolutional procedure [35] can be expressed as:

$$\begin{aligned}
\boldsymbol{X}_t^{out} &= \sigma\left(\widehat{\boldsymbol{A}}\,\mathrm{Relu}\left(\widehat{\boldsymbol{A}}\boldsymbol{X}_t^{road}\boldsymbol{W}_1\right)\boldsymbol{W}_2\right), \\
\widehat{\boldsymbol{A}} &= \widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}, \\
\widetilde{\boldsymbol{A}} &= \boldsymbol{A} + \boldsymbol{I}_N,
\end{aligned} \tag{19}$$

where adjacency matrix $\boldsymbol{A}$ is used to denote the edge connectivity of each road, and the element of this matrix equals 1 if and only if the two roads are connected. $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ are the training parameter of two graph convolutional layers, $\boldsymbol{X}_t^{road}$ is the road features matrix in time slot t, $\boldsymbol{X}_t^{out}$ represents the feature matrix in time slot t, $\widetilde{\boldsymbol{D}}$ is a degree matrix, and $\sigma(\cdot)$, $\mathrm{Relu}(\cdot)$ represent the activation function.

C. Spatio-Temporal Features Fusion

As aforementioned, three feature vectors from the temporal and spatial aspects can be obtained. To fuse spatio-temporal features, each vector will be transformed into a one-dimension vector by passing a flatten layer. Then three one-dimension vectors will be concatenated into a vector and to be the input of the Spatio-temporal feature fusion network.

When making migration decisions, we care about the number of vehicles that have changed their accessed cell. Thus, the prediction output is a 1×7 vector, whose element denotes the probability of the vehicle entering the six adjacent cells or staying in the current cell.

The probability vector will be converted into the probability vector $\boldsymbol{p}_s$ by passing a soft-max layer.

$$Loss = -\sum_{i=1}^{m} \log \frac{\exp\left(\boldsymbol{p}_{si}\right)}{\exp\left(\sum_{i=1}^{m} \boldsymbol{p}_{si}\right)} \boldsymbol{y}_i, \tag{20}$$

Algorithm 1: The Training Process of Spatio-Temporal Mobility Prediction Networks.

Input: $\boldsymbol{X}^{drive} = (\boldsymbol{x}_t^{drive}, \boldsymbol{x}_{t-1}^{drive}, \ldots, \boldsymbol{x}_{t-\tau}^{drive})$,
Output: $\boldsymbol{p}_s$
1: Collect $\boldsymbol{X}^{trajectory}$ and $\boldsymbol{X}^{road}$ from edge server database based on target vehicle data $\boldsymbol{X}^{drive}$.
2: $epoch \leftarrow 0$
3: **repeat**
4: $epoch \leftarrow epoch + 1$
5: $batch \leftarrow 0$
6: **repeat**
7: $batch \leftarrow batch + 1$.
8: Extract the temporal features from $\boldsymbol{X}^{drive}$ and $\boldsymbol{X}^{trajectory}$ based on (18).
9: Extract the spacial feature from $\boldsymbol{X}^{road}$ based on (19).
10: Concatenate three features as a spatio-temporal feature vector and get the prediction results.
11: Calculate the loss based on loss function (20), and back-propagate the loss value.
12: **until** $batch \geq \lceil |Epoch|/256 \rceil$
13: **until** $epoch \geq 500$

where $\boldsymbol{y}$ is the label value, which is constructed as a one-hot code.

The whole training process is shown in Algorithm 1. We set the batch size to 256. In every epoch, the training loss will be calculated based on (20). The loss value will be back-propagated to the parameters of each layer. After the training process, the maximum value of $\boldsymbol{p}_s$ will be chosen as the prediction result. For the online prediction, we follow the online prediction offline training principles. The training process is conducted offline at the beginning to get the initial model parameters for online prediction. Then the trained neural parameters will be updated periodically based on the collected data, and the model parameters will be optimized by Algorithm 1 during the whole simulation.

In the end, we can obtain which cell each vehicle is expected to arrive in the next time slot. According to the current cell and predicted cell of vehicles, we can calculate the vehicle arrival rate $\alpha_i(t+1)$ and the vehicle departure rate $\beta_i(t+1)$ of any server s_i. Based on the vehicle arrival rate and departure rate, we can further estimate the workload of servers and make optimal migration decisions.

V. Proactive Service Migration

In this section, we focus on making optimal migration decisions to minimize energy consumption and ensure the QoS of each vehicle. Based on the obtained vehicle arrival rate $\alpha_i(t+1)$ and departure rate $\beta_i(t+1)$, we can calculate the arrival rate of vehicles requesting edge services $\gamma_i(t+1)$ using (10). Then, according to (11), the queue length $Q_i(t+1)$ can be derived.

In this paper, the server workload is formulated as a queue system. The Lyapunov optimization is a suitable method to solve the objective function in a stable queue [36] [37]. Considering the queue length $Q_i(t)$, the Lyapunov optimization method is used to solve the formulated problem.

First, based on (11), the Lyapunov function is defined as

$$L(t) = \frac{1}{2}(Q_i(t))^2. \tag{21}$$

Then one-slot Lyapunov drift can be given as

$$\Delta(Q_i(t)) = \mathbb{E}[L(t+1) - L(t)|Q_i(t)]. \tag{22}$$

Generally, servers are intended to host as many services as possible to improve efficiency. However, increasing service numbers will lead to server overload. To maximize $G(M(t))$ and avoid server work on the overload state, the drift-plus penalty function can be defined as

$$\begin{aligned} &\min_{M} \sum_{s_i \in S} \sum_{v \in s_i} \Delta\left(Q_i(t)\right) - V\mathbb{E}[G(M(t))|Q_i(t)], \\ &\quad \text{s.t. (a) } \delta(t) \in \{0,1\}, \forall t, \\ &\qquad \text{(b) } \varphi(t) \in \{1,2,\cdots m\}, \forall t, \\ &\qquad \text{(c) } N_i \leq N_{\max}, \forall s_i, t, \end{aligned} \tag{23}$$

where V is the parameter to balance the different quantities of two parts. Constraint (a) (b) ensures that migration decisions $M(t) = (\delta(t), \varphi(t))$ are within the feasible domain. Constraint (c) is used to prevent server work in an overload state.

Minimizing the drift-plus penalty function is equivalent to minimizing the upper bound of the drift-plus penalty function. Based on the upper bound of the drift-plus penalty function, the objective function (23) can be simplified as

$$\begin{aligned} &\min_{M} \sum_{s_i \in S} \sum_{v \in s_i} \mathbb{E}[Q_i(t)\left(\gamma_i(t+1) - N_i\right) - VG(M(t))|Q_i(t)], \\ &\quad \text{s.t. (a) } \delta(t) \in \{0,1\}, \forall t, \\ &\qquad \text{(b) } \varphi(t) \in \{1,2,\cdots m\}, \forall t, \\ &\qquad \text{(c) } N_i \leq N_{\max}, \forall s_i, t. \end{aligned} \tag{24}$$

Proof can be seen in Appendix-B, available in the online supplemental material.

To solve this problem, we need to remove the expectation from the formulated problem first. Since the queue length $Q_i(t)$ only depends on the arrival rate and leaving rate of server s_i in the previous time step, the formulated problem can be separated into per-server problems. Also, $Q_i(t)$ at the current time step t is deterministic, so the condition of $Q_i(t)$ can be removed as

$$\begin{aligned} &\min_{M} \sum_{v \in s_i} \mathbb{E}[Q_i(t)\left(\gamma_i(t+1) - N_i\right) - VG(M(t))], \\ &\quad \text{s.t. (a) } \delta(t) \in \{0,1\}, \forall t, \\ &\qquad \text{(b) } \varphi(t) \in \{1,2,\cdots m\}, \forall t, \\ &\qquad \text{(c) } N_i \leq N_{\max}, \forall s_i, t. \end{aligned} \tag{25}$$

In each time step, each vehicle has the probability p_g to generate a new service request, the service arrival number follows

Algorithm 2: Lyapunov Optimization-Based Online Proactive Service Migration Method.

Input: $Q_i(t)$, $\boldsymbol{p}_s$
Output: $M(t) = (\delta(t), \varphi(t))$
1: Based on the mobility prediction result $\boldsymbol{p}_s$, select the migration target server $s_{\varphi(t)}$.
2: Calculate the $Q_i(t+1)$ for every server based on vehicle prediction results and the (11).
3: **for** $i = 1$ to m **do**
4: **for** $v \in s_i$ **do**
5: Get migration decision $(\delta(t), \varphi(t))$ by solving (24).
6: **if** $\delta(t) = 1$ **and** $s_{\varphi(t)}$ overloaded **then**
7: Select the next closest non-overloaded server of $s_{\varphi(t)}$ and back to Step 5.
8: **end if**
9: **end for**
10: **end for**

the binomial distribution with expectation equal to $\gamma_i(t+1)$ given in (10). In (25), only $\gamma_i(t+1)$ is related to the stochastic variable, so the expectation can be calculated as

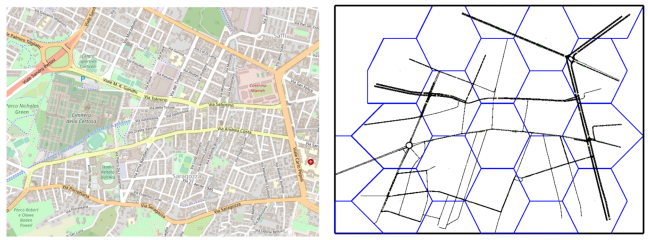
$$\begin{aligned} &\min_{M} \sum_{v \in s_i} Q_i(t)\left(\gamma_i(t+1) - N_i\right) - VG(M(t)), \\ &\text{s.t. (a) } \delta(t) \in \{0,1\}, \forall t, \\ &\quad \text{(b) } \varphi(t) \in \{1,2,\cdots m\}, \forall t, \\ &\quad \text{(c) } N_i \leq N_{\max}, \forall s_i, t. \end{aligned} \tag{26}$$

Then the formulated optimization problem becomes an integer programming problem, and we solve it by using the implicit enumeration method.

In summary, the whole proactive service migration process is given in Algorithm 2. First, $Q_i(t+1)$ and $\gamma_i(t+1)$ for each server are calculated based on the mobility prediction results using (9) and (10), and $Q_i(t+1)$ is used to evaluate whether the migration will cause any server overload in the next time step. Then, the implicit enumeration method is used for solving the formulated integer programming problem in (26). After that, $Q_i(t+1)$ is updated based on the obtained solution. If any migration could lead to server overload, this migration will be blocked and the selected migration servers will be removed from the feasible domain at (26b). Then we will solve the problem again in the updated feasible domain until a feasible solution is found. In this way, server overload during migration can be mitigated.

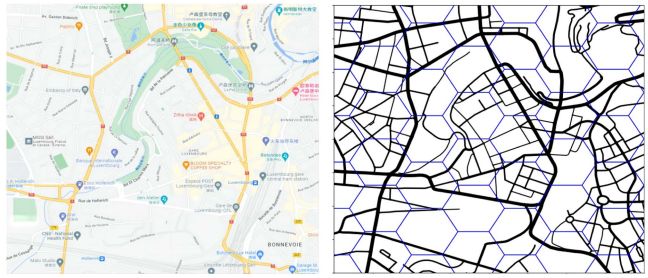
VI. Simulation

In this section, extensive simulations are conducted to verify the superiority of the proposed proactive migration method. First, we compare the mobility prediction accuracy with Double GRU, GCN, LSTM, GRU+CNN, and TCP methods. Then, we compare the migration performance in terms of service drop rate, energy consumption, and queue length of the edge server.



(a) Bologna city map (b) Road structure in SUMO

Fig. 5. The Real-World Bologna simulation SUMO scenario includes a part of the inner city of Bologna (1500 × 1800 square meters). The whole area is divided into 21 cells, each cell occupying 200 × 200 square meters. (a) The real-world map of the simulation area. (b) The SUMO simulation scenario.



(a) Luxembourg city map (b) Road structure in SUMO

Fig. 6. The Luxembourg simulation SUMO scenario includes a part of the inner city of Luxembourg (2000 × 3000 square meters). The whole area is divided into 52 cells, each cell occupying 200 × 200 square meters. (a) The real-world map of the simulation area. (b) The SUMO simulation scenario.

A. Dataset and Network Settings

1) Dataset Setting: In this paper, two real-world datasets are used to evaluate the performance of the designed neural network. The first dataset, "Real-World Bologna" [38] contains 2,344,894 data samples taken from 8,779 vehicles during rush hour on 11-13 of 2008. The Bologna scenario includes a part of the inner city of Bologna (1500 × 1800 square meters), Italy, which includes the whole area between the two main streets Andrea Costa and Pasubio shown in Fig. 5(a). Based on the real-world map, the whole area is divided into 21 cells, and each cell occupies 200 × 200 square meters shown in Fig. 5(b).

The second dataset, "Luxembourg SUMO Traffic" [39], is larger than the Bologna dataset, and it contains 29,526,813 data samples taken from 47,152 vehicles during rush hour and covers more than 4,000 intersections. The raw traffic demand is collected from the internet site of the Luxembourg National Institute of Statistics and Economic studies. Based on the real-world map, the downtown area is selected, including the inner city of Luxembourg (2000 × 3000 square meters) shown in Fig. 6(a). The whole area is divided into 52 cells, and each cell occupies 200 × 200 square meters shown in Fig. 6(b).

We conducted two case studies on the Bologna dataset and the Luxembourg dataset. Each dataset is divided into three subsets according to the vehicles: 70% of vehicles are used as training data, 10% of vehicles are used as validation data, and the rest 20% of vehicles are used as testing data.

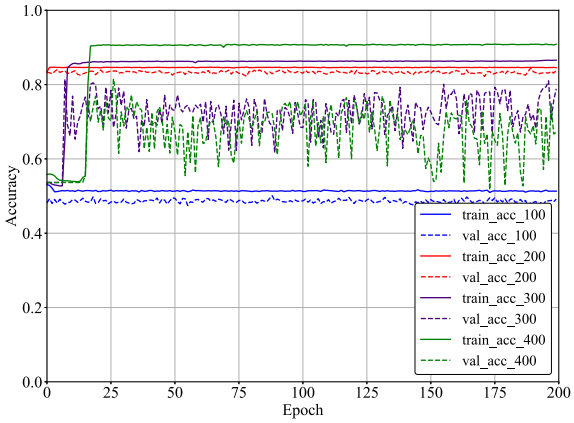


Fig. 7. The accuracy comparison of cell size under 100×100 m^2, 200×200 m^2, 300×300 m^2, and 400×400 m^2.

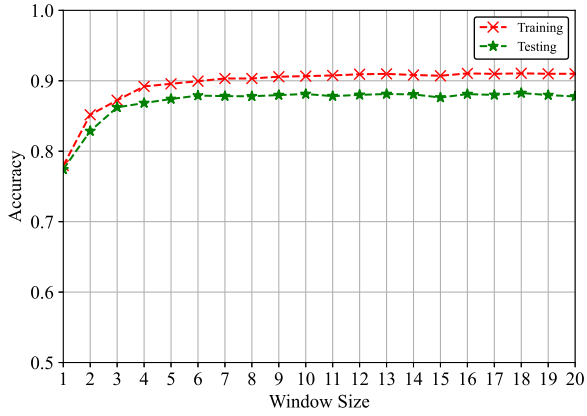


Fig. 8. The accuracy comparison under different window sizes from 1 to 20.

2) Parameters Setting: In this subsection, several simulation parameters including the cell size, window size, and utility function will be discussed.

The cell size could affect the accuracy of the prediction results. We chose four cell sizes from 100×100 m^2 to 400×400 m^2 mentioned in [40] [41] for selecting a suitable cell size.

As shown in Fig. 7, we verify the training and validation accuracy of the proposed method in four different cell sizes. In the case of 100×100 m^2, the output state number needs to be increased from 7 to 13 since the vehicle could go beyond the adjacent cells. The increased number of output states leads to model underfitting. In the cases of 300×300 m^2 and 400×400 m^2, the training label of the dataset is somewhat imbalanced due to the large cell size, which makes the vehicle barely change the cell. Although their accuracy improved on the training set, the validation accuracy decreased due to model overfitting. Among the four cases, the size of 200×200 m^2 performed best, so we choose it as the cell size in the subsequent simulation experiments. In a practical system, if the real cell size is substantially smaller than our setting, the duration of the time slot could be reduced accordingly. If the real cell size is much larger, it can be divided into smaller sub-cells and predict which sub-cell the vehicle will be located in the next time slot.

The window size τ is defined to decide how long we want to learn the vehicle feature from the past. Different window sizes could affect the accuracy of the prediction results. We select the window size from 1 to 20 to find a suitable window size.

As shown in Fig. 8, the accuracy of the proposed mobility prediction method under different window sizes is verified. Intuitively, the most recent temporal features are beneficial for improving prediction accuracy. So both the training accuracy and testing accuracy increase significantly when the window size increases from 1 to 5. Then, we further increase the window size from 5 to 20, the improvement is not obvious compared to that from 1 to 5. Considering the larger window size leads to higher computation complexity, the window size of 5 is selected in the subsequent simulation experiments.

3) Neural Network Setting: The architecture of the designed neural network is given in Table II. For driving features extraction GRU, each layer has 8 GRU cells. Each layer's final state h_n will be the output of the two-layers GRU part. For historical trajectory extraction GRU, each layer has 8 GRU cells, and the output is the final state h_n of the second layer.

TABLE II
THE ARCHITECTURE OF NEURAL NETWORK

	Layer	Parameter	
Driving Features Extraction Network		Input Size	Hidden Size
	GRU1	4	8
	GRU2	4	8
	Drop-Out (p=0.3) & Flatten		
Historical Trajectory Extraction Network		Input Size	Hidden Size
	GRU1	4	8
	GRU2	4	8
	Drop-Out (p=0.3) & Flatten		
Road Features Extraction Network		Input Channel	Out Channel
	GCN1	25	5
	GCN2	5	1
	Drop-Out (p=0.3) & Flatten		
Feature Fusion Linear Network		Input Feature	Out Feature
	Linear1	29	10
	Linear2	10	7
	Log-softmax		

To learn the future traffic trend from adjacent road sections, the input channel of the first GCN is set as 25, which consists of 5-slots road features from the current road section and 4 adjacent road sections. The output channel number is set equal to the number of road sections. The input channel number of the second GCN layer is 5, and the output channel number is 1. The fusion network contains two fully connected neural layers. The neural number of the first layers is equal to the element number of the spatio-temporal vector. Then the 1×7 vector will be converted into the probability vector $\boldsymbol{p}_s$ by passing a log-softmax layer.

The hyperparameters of the training process are given in Table III. The two datasets share the same training hyperparameters. The window size τ is set to 5, and the historical window size is set to 100. The batch size is set at 256, and the drop-out probability is set at 0.3. Due to the different scales of the two datasets, the training epoch is set to 500 for the Bologna dataset and 200 for the Luxembourg dataset. To avoid over-fitting, we use the step-learning rate to decay the learning rate every 50 epochs with a decay rate of 0.9.

TABLE III
THE HYPERPARAMETERS SETTING

Parameter	Value
Window size τ	5
Historical window size l	100
Training epoch	200/500
Batch size	256
Learning rate	1e-3
Optimizer	Adam
StepLR decay	50 epoch
StepLR gamma	0.9
Drop-out probability	0.3

TABLE IV
THE RAW DATASET FEATURES

Features Type	Features Name	Description
Time	timestep	The simulation time step
Vehicle	V_ID	The ID of the vehicle
	Lane	The ID of the current lane
	Azimuth	The azimuth of the vehicle
	Speed	The speed of the vehicle
	Pos_x	The absolute X coordinate of the vehicle
	Pos_y	The absolute Y coordinate of the vehicle
Road	R_ID	The ID of the road lane
	Travel time	The mean travel time on the lane
	Maxspeed	The maximum speed of the vehicles on the lane
	Meanspeed	The mean speed of the vehicles on the lane
	Occupancy	The occupancy percentage of the lane

B. Dataset Pre-Processing

The dataset used in this paper includes the vehicle features and the road traffic features in each time slot. As shown in Table IV, the raw vehicle features consist of the vehicle ID, current road lane ID, speed, azimuth, and coordinate of this vehicle. The raw road features consist of the road lane ID, the mean travel time, mean speed, max speed, and the percentage of lane occupancy.

To fully extract the hidden correlation from raw features, the raw dataset is converted into three parts shown in Table V. For the target vehicle, driving data are converted into τ-length driving data $\boldsymbol{X}^{drive} = (\boldsymbol{x}_t^{drive}, \boldsymbol{x}_{t-1}^{drive}, \ldots, \boldsymbol{x}_{t-\tau}^{drive})$. Each driving data $\boldsymbol{x}_t^{drive}$ contains the speed, azimuth, and coordinate of the vehicle at time slot t. Then, the sequential $\boldsymbol{X}^{drive}$ is taken as the input of the first GRU layer of the driving features extraction network.

In addition to the features of the target vehicle, the hidden correlation from the other vehicles that have the same trajectory as the target vehicle is also beneficial for mobility prediction. Similar to the driving data, a window with length l is set to convert the trajectory into τ-length data as $\boldsymbol{X}^{trajectory} = (\boldsymbol{x}_{t-l}^{trajectory}, \ldots, \boldsymbol{x}_{t-l-\tau}^{trajectory})$, and $\boldsymbol{X}_{t-l}^{trajectory}$ contains the speed, azimuth, and coordinate of the vehicle at time slot $t-l$. Then, $\boldsymbol{X}^{trajectory}$ is taken as the input of the historical trajectory extraction network.

To extract the traffic trend from adjacent road sections, the current road sections of the target vehicle and four adjacent road sections are converted as a graph G. Each vertex in G denotes a road section, which contains four road features. They are the mean travel time, mean speed, max speed, and the percentage of lane occupancy. Each edge in G denotes the connectivity of each road. Then graph G is the input of the road features extraction network.

C. Performance of Mobility Prediction

Performance Metric: In addition to the correct rate, we further divide the errors into three categories: premature-prediction errors (PE), late-prediction errors (LE), and mis-prediction errors (ME), which correspond to three types of migration errors. The detailed descriptions are given as follows:

- Correct prediction (CP): The predicted cell is the vehicle-accessed cell at time slot $t+1$. The prediction is correct.
- Premature-prediction error (PE): The predicted cell is the next accessed cell in the future. The prediction is wrong.
- Late-prediction error (LE): The predicted cell is the current cell, but the vehicle will change the accessed cell in the next time slot. The prediction is wrong.
- Mis-prediction error (ME): The predicted cell is neither the next accessed cell in the future nor the previous cell in the past. The prediction is wrong completely.

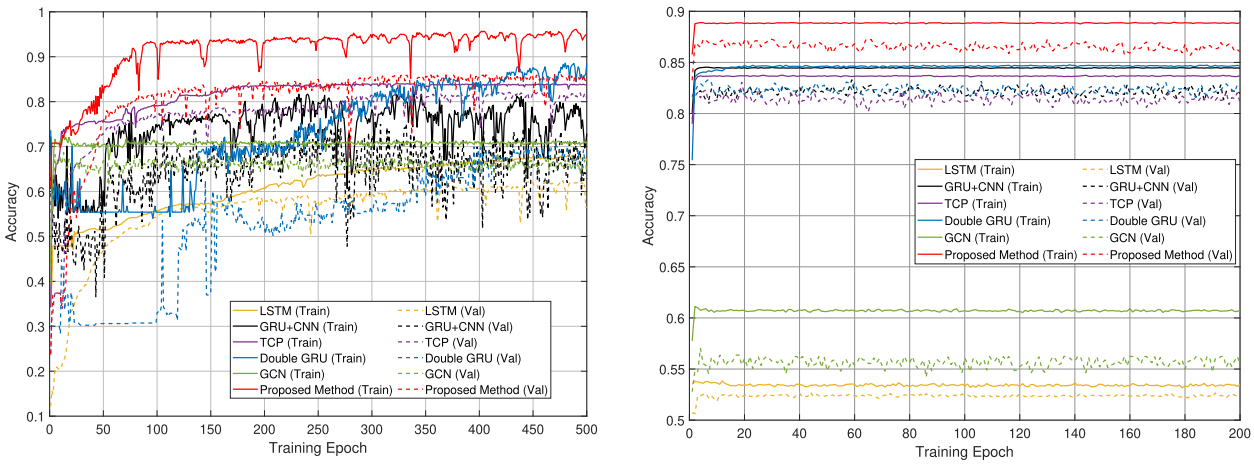
Compared Methods: To evaluate the performance of the proposed spatial-temporal mobility prediction method, we compare it with the following three state-of-the-art methods and two baseline methods.

- LSTM: a time series based prediction method adopted in [22], which predicts the user density in a specific cell by using two LSTM layers.
- GRU+CNN: a multi-feature fusion method designed in [24], which extracts features from the driving data and the sequence of visited cells. The mobility prediction method consists of two GRU layers and two convolution layers.
- TCP: a traffic context prediction method proposed in [42], which predicts accessed cells by combining 4 different candidate prediction algorithms. The candidate prediction algorithms include two statistical-based algorithms, simple average and Prophet, and two learning-based algorithms, Auto-Arima and LSTM. The best performance among the candidate algorithms is used as the prediction result.
- Double GRU: a baseline method predicts accessed cells using drive-based data and historical trajectory data, which is the temporal part of the proposed method and just uses the temporal features.
- GCN: a baseline method predicts accessed cells by using road-based traffic data, which is the spatial part of the proposed method and just uses the spatial features.

1) Case study1: Bologna Dataset: As shown in Fig. 9(a), the proposed method has the highest training accuracy. LSTM and GCN methods that use only one type of spatial or temporal feature can achieve an accuracy of 69% at most. In contrast, for the Double-GRU and GRU+CNN methods combining multiple types of features, the accuracy can be further increased to 86%. Since the TCP method incorporates four different prediction methods, it can learn the hidden dependencies from different aspects, and its accuracy can also reach 83%. Combining spatial and multi-scales temporal features, the proposed GRU+GCN

TABLE V
THE INPUT FEATURES OF SPATIO-TEMPORAL NEURAL NETWORKS

	Data Source	Input Size	Features Name	Description
Real-Time Driving Data	Target Vehicle	$4 \times \tau$	Azimuth	The azimuth of the target vehicle
			Speed	The speed of the target vehicle
			Pos_x	The absolute X coordinate of the target vehicle
			Pos_y	The absolute Y coordinate of the target vehicle
Historical Trajectory	Historical Vehicle	$4 \times \tau$	Azimuth	The azimuth of the historical vehicle
			Speed	The speed of the historical vehicle
			Pos_x	The absolute X coordinate of the historical vehicle
			Pos_y	The absolute Y coordinate of the historical vehicle
Road-Based Traffic Data	Current & Adjacent Road Sections	$4 \times 5 \times \tau$	Travel time	The mean travel time on the specific lane
			Maxspeed	The maximum speed of the vehicles on the specific lane
			Meanspeed	The mean speed of the vehicles on the specific lane
			Occupancy	The occupancy percentage of the lane



(a) Bologna Dataset (b) Luxembourg Dataset

Fig. 9. Training and validation accuracy comparison during the training procedure in the Bologna dataset and the Luxembourg dataset. (a) The accuracy during the 500 training epoch in the Bologna dataset. (b) The accuracy during the 200 training epoch in the Luxembourg dataset.

TABLE VI
COMPARISON PERFORMANCE ON PREDICTION ACCURACY

Dataset	Method	Training Set				Validation Set				Testing Set			
		CP	PE	LE	ME	CP	PE	LE	ME	CP	PE	LE	ME
Bologna	LSTM	68.06%	6.59%	1.05%	24.30%	62.39%	6.29%	1.31%	30.01%	61.89%	6.67%	1.21%	30.23%
	GRU+CNN	80.69%	0.06%	0.02%	19.22%	75.21%	0.17%	0.02%	24.60%	78.34%	0.06%	0.02%	21.58%
	TCP	83.71%	8.41%	2.91%	4.97%	82.03%	9.37%	3.81%	4.79%	82.33%	9.03%	4.18%	4.46%
	Double-GRU	86.06%	5.09%	0.40%	8.42%	70.07%	5.88%	0.82%	23.21%	71.06%	1.70%	1.01%	26.22%
	GCN	69.43%	8.67%	1.07%	20.82%	68.68%	9.15%	1.44%	20.71%	69.23%	9.55%	1.30%	19.90%
	Proposed Method	92.82%	1.98%	0.03%	5.17%	86.06%	5.04%	0.09%	8.79%	88.32%	4.79%	0.08%	6.79%
Luxembourg	LSTM	53.50%	13.98%	9.40%	23.12%	52.09%	11.32%	14.94%	21.65%	51.43%	12.23%	16.06%	20.28%
	GRU+CNN	84.18%	8.26%	2.59%	4.97%	83.31%	8.52%	3.65%	4.52%	82.66%	8.76%	3.88%	4.70%
	TCP	83.62%	8.44%	2.96%	4.98%	83.08%	8.57%	3.78%	4.57%	82.93%	8.94%	3.73%	4.40%
	Double GRU	84.61%	7.49%	3.39%	4.51%	83.90%	8.24%	4.06%	3.80%	83.61%	8.22%	4.58%	3.59%
	GCN	60.92%	10.42%	4.13%	24.53%	57.03%	10.32%	3.12%	29.53%	61.25%	11.49%	4.52%	22.74%
	Proposed Method	88.89%	5.31%	2.54%	3.26%	87.25%	4.46%	3.67%	4.62%	87.52%	5.23%	3.16%	4.09%

method has the highest accuracy among other methods, reaching 92%.

The prediction results of the validation set and the testing set are given in Table VI. According to the testing results, the Double-GRU method is seriously overfitting, and the accuracy rate is reduced severely from 86% to 70%. The accuracy of LSTM, GRU+CNN, and the proposed method also decreases. Nevertheless, the accuracy of the proposed method is the highest among all methods. For the correct prediction rate, the accuracy of the proposed method can achieve 92.82% in the training set, 86.06% in the validation set, and 88.32% in the test set, at least 6% higher than other methods.

2) Case study2: Luxembourg Dataset: The training accuracy of six prediction methods in the Luxembourg dataset is given in Fig. 9(b). Compared to the evaluation in the Bologna dataset, the evaluation results in the Luxembourg dataset are more stable during the training process. The reason causes the difference is that the Luxembourg dataset is almost 11 times larger than the Bologna dataset. Data samples in the small dataset have a higher variance between each other than data samples in the

large dataset, which increases the variance in prediction results and causes more fluctuations during the training process [43]. To avoid the impact caused by the dataset size, a large-scale dataset in Luxembourg is used for model training.

Similar to the performance comparison in the Bologna dataset, LSTM and GCN methods that use only one type of spatial or temporal feature have the lowest accuracy of 61%. The TCP and GRU+CNN methods that consider more than one type of feature or incorporate different prediction methods will get higher accuracy of 83%. Among the six methods, the proposed method has the highest accuracy of 88%. Since the Luxembourg dataset is almost 11 times larger than the Bologna dataset, all six methods can converge within 5 training epochs.

The detailed prediction results in the validation set and the testing set are given in Table VI. In the large-scale dataset, overfitting is alleviated. All six methods achieve similar accuracy on the training, validation, and testing set. The LSTM method obtains the lowest accuracy of 51.43% on the testing set and 52.09% on the validation set, followed by the GCN method which gets an accuracy of 61.25 on the testing set and 57.03% on the validation set. The results of LSTM and GCN show that one type of feature is not sufficient for predicting mobility accurately. The multi-feature method, such as Double-GRU, GRU+CNN, and TCP can achieve higher accuracy on the testing set. Considering both spatial and multi-scales temporal features, the proposed method achieves the highest accuracy of 88.89% on the training set, 87.25% on the validation set, and 87.52% on the testing set.

D. Performance of Service Migration

In this subsection, the simulation settings used to evaluate the performance of the migration method are introduced. The number of MEC servers is set to match the number of cells in each case. The total time step number is 5000 corresponding to the SUMO simulation length and the duration of each time step is 5 seconds. The cellular size is 200×200 m^2 as discussed in 6.1.2. The wireless link between the vehicles and the BS is using the NLOS mm-wave channel model of [44] under the 802.11bd protocol standard [45]. The maximum speed of the road is from 13.89 m/s to 19.44 m/s based on the real-world road speed limits in the city center. The average speed of vehicles varies from 0 m/s to 19.44 m/s in each time step based on the traffic condition. In this simulation, the service number of vehicles in each time step varies from 0 to 696, the block number varies from 0 to 381, and the energy consumption varies from 9.73×10^6 J to 1.39×10^{10} J. For quantity normalization, the maximum value in each quantity is selected, and the weights of the utility functions are set to 0.0014, 0.0026, and 7.1712×10^{-9} corresponding to the reciprocal of them. In the practical system, these weights can be flexibly set to different values to trade off the importance of the three terms in the utility function according to the practical requirement. Other settings include transmit power, transmit rate, and capacity referred to the related migration studies [23] [24], and the detailed simulation setting is given in Table VII.

To meet the requirements of connected vehicles in practical networks, three fundamental services are considered in this paper. Three types of services correspond to three different network traffic conditions: (1) Low upload traffic and low download traffic; (2) Low upload traffic and high download traffic; (3) High upload traffic and low download traffic. The upload traffic data (UL) and download traffic data (DL) are collected from the real-world mobile cellular traffic scenario under the topology of the user-to-data center model in [46]. The detailed settings are shown in Table VIII.

TABLE VII
SIMULATION SETTINGS

Parameter	Value
The number of MEC server m	21/52
The cellular size	200×200 m^2
The interval of a time step t	5 s
The total time step number	5000
The average speed of vehicles	[0, 19.44] m/s
Maximum speed of vehicles	[13.89, 19.44] m/s
Maximum transmit power of vehicles P_v	0.2 W
Maximum transmit power of edge server P_s	320 W
Maximum transmit rate R	1180 Mb/s
Maximum capacity of edge server C_L	50
Quantity normalization weight w_1	0.0014
Quantity normalization weight w_2	0.0026
Quantity normalization weight w_3	7.1712×10^{-9}
Host connection revenue χ	2
Blocking new services loss χ_{bn}	0.5
Blocking migration services loss χ_{bm}	0.1

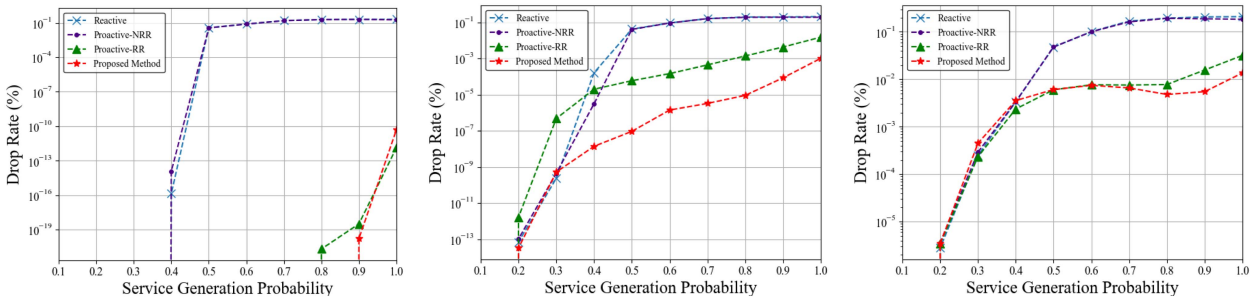
TABLE VIII
SERVICES SETTINGS FOR PERFORMANCE COMPARISON

Service Type	UL Traffic c_{up} (kB/s)	DL Traffic c_{down} (kB/s)	On-State Probability p_{on}
Navigation	184.32	440.32	0.1
Data sharing	194.56	5089.28	0.3
Data uploading	2580.48	92.16	0.5

We select the previously mentioned two types of proactive service migration methods as the compared method, the LSTM-based method and the GRU+CNN-based method respectively. The Double-GRU and GCN methods are excluded since those two prediction methods are part of our method. In addition, a reactive method is added to the simulation as the baseline method.

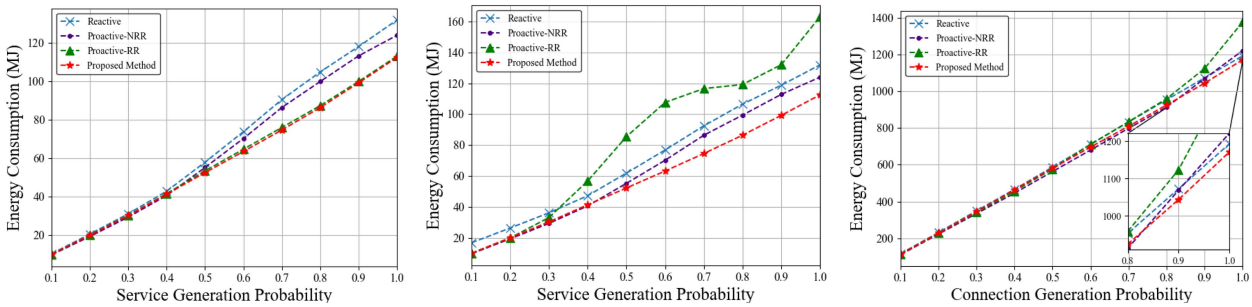
For consistency and clarity, the migration methods are denoted by their characteristics. The description of the three compared migration methods is given as follows:

- Reactive: a baseline reactive service migration method comes from [17] that always migrates services to the access server.
- Proactive-NRR: a proactive service migration method is adopted in [22], which does not consider resource reservations. The two-layer LSTM is used to make mobility predictions, and the prediction result is used to determine the migration destination of the services.
- Proactive-RR: a mobility-aware dynamic migration method is designed in [24]. The GRU+CNN is used to make mobility predictions. The prediction result is used to determine which server to migrate to and if the target server does not meet the resource reservation requirement, the server with the next highest possibility of prediction will be selected.



(a) Navigation (b) Data sharing (c) Data uploading

Fig. 10. The performance comparison of service drop rate under three scenarios, navigation, data sharing, and data uploading respectively. (a) The service drop rate for navigation, which has low UL and DL traffic, and low on-state probability $p_{on} = 0.1$; (b) The service drop rate for data sharing, which has low UL traffic, high DL traffic, and middle on-state probability $p_{on} = 0.3$; (c) The service drop rate for data uploading, which has high UL traffic, low DL traffic, and high on-state probability $p_{on} = 0.5$. The results show that the proposed method has the lowest service drop rate in almost all scenarios.



(a) Navigation (b) Data sharing (c) Data uploading

Fig. 11. The performance comparison of energy consumption under three scenarios, navigation, data sharing, and data uploading respectively. (a) The energy consumption for navigation, which has low UL and DL traffic, and low on-state probability $p_{on} = 0.1$; (b) The energy consumption for data sharing, which has low UL traffic, high DL traffic, and middle on-state probability $p_{on} = 0.3$; (c) The energy consumption for data uploading, which has high UL traffic, low DL traffic, and high on-state probability $p_{on} = 0.5$. The results show that the proposed method can save up to 10% of energy compared to other state-of-art methods.

First, the performance comparison of drop rate under three types of services is shown in Fig. 10. As shown in Fig. 10(a), the proposed method rarely drops connections under low p_{on} condition. When $p_g \leq 0.5$, the drop rate of the Proactive-NRR and the Reactive method increases to 0.1, which indicates part of the servers are overloaded. By contrast, the proposed method and Proactive-RR have the drop rate at or below 10^{-10} even when $p_g = 1$. Both are sufficiently low.

In Fig. 10(b), a higher p_{on} are considered. When p_{on} increases, more services need to be hosted by edge servers simultaneously. The high drop rate is observed in Proactive-NRR and Reactive methods since all the services are migrated close to their locations, which leads to server overload in high-dense traffic areas. With the consideration of resource reservation, the proposed method can reduce the drop rate significantly. The Proactive-RR method gives a dropping rate of 1.51% when $p_g = 1$, while the proposed method only gives 0.13% in the same condition. Thanks to the more accurate prediction, the proposed migration method remains to have the lowest drop rate of 1.36% even when $p_{on} = 0.5$ in Fig. 10(c). It is almost a third of Proactive-RR's dropping rate of 3.18% in the same condition.

The performance of energy consumption is given in Fig. 11. Considering the highest connection generation rate in Fig. 11(a), our proposed method has almost the lowest energy consumption compared to other methods. Due to the high service dropping rate and misprediction possibility, the Reactive and Proactive-NRR methods need to retransmit dropped requests in the previous time slot, and the energy consumption of those two methods is relatively high.

A significant energy consumption gap occurred in the high download case shown in Fig. 11(b). In high download traffic cases, energy consumption increases with communication links. Because the Proactive-NRR method and Reactive method always migrate services to the closest server, the energy consumption of those two methods is relatively low.

However, the resource reservation considered method will block some migrations to maintain a relatively low service drop rate. Due to the high misprediction rate of the Proactive-RR method, the blocked services may be migrated to a distant server, so the highest energy consumption comes from the Proactive-RR method.

In contrast, our online proactive migration algorithm will choose the server closest to the predicted server instead of the second possible server when the migration request is blocked. It can reduce the risk of migrating services to a remote server. Therefore, our methods can achieve the lowest energy consumption at 112.46 MJ, which is almost 10% less than the second-lowest method at 123.90 MJ.

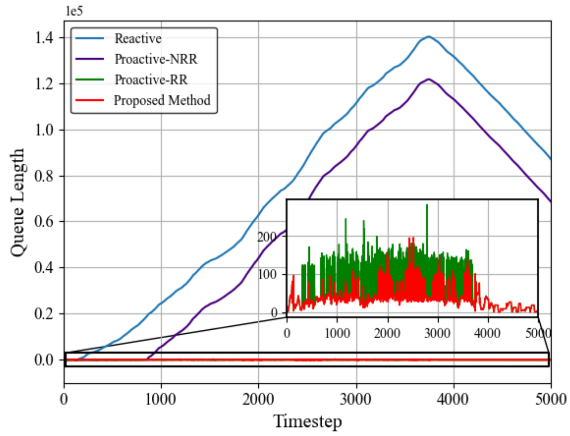


Fig. 12. The performance comparison of queue length of server. The queue is on the server with the highest workload among 21 servers during the whole simulation in the 5000-time steps. The worst case is considered by setting $p_g = 1$ and $p_{on} = 0.5$. The results show that the proposed method can maintain the shortest queue even in the worst case.

The high upload traffic case is shown in Fig. 11(c). The gap is not obvious compared to the high download case since the transmission power of the vehicle is relatively small. Accordingly, the energy consumption of different methods is close. But if we focus on the largest generation probability cases, the proposed method still spends the least energy at 1170.22 MJ, which has saved 2% compared to the second-lowest method at 1191.95 MJ.

Finally, we verify the real-time performance by illustrating the queue length of servers. The queue length of the server with the highest workload servers during the whole simulation in the 5000-time step is shown in Fig. 12. For the Reactive and Proactive-NRR methods that ignored the resource reservation, the queue length is increasing to 140000, which means the delay will become unbounded. By contrast, the proposed method can maintain a relatively small queue length by considering resource reservation. Thanks to the accurate spatio-temporal prediction, the proposed method can maintain the queue length of 60 in the worst case, which is almost half of the Proactive-RR method in the same condition.

VII. Conclusion

In this paper, a spatial-temporal mobility awareness proactive service migration method is proposed to make migration decisions based on the mobility of vehicles. Empowered by spatial-temporal mobility prediction, the optimal migration decision can be made to improve the QoS and reduce energy consumption. The extensive experiments validate the superiority of the proposed spatial-temporal mobility awareness proactive service migration method in prediction accuracy and migration performance. The results show that the proposed method has the highest accuracy among other methods and improved prediction accuracy by 6% in two real-world datasets. In terms of service drop rate, our method has one-third of the best-compared method in the worst case. For energy consumption, the proposed method can save up to 10% of energy and maintain the shortest queue length in all cases.

We anticipate the presented method would offer insights and opportunities to migrate services proactively. For future extensions, more real-world datasets and more types of services will be considered.

References

[1] J. Hu, C. Chen, L. Cai, M. R. Khosravi, Q. Pei, and S. Wan, "UAV-assisted vehicular edge computing for the 6G Internet of Vehicles: Architecture, intelligence, and challenges," *IEEE Commun. Standards Mag.*, vol. 5, no. 2, pp. 12–18, Jun. 2021.

[2] Y. Ni, C. Zhao, and L. Cai, "Hybrid RSU management in cybertwin-IoV for temporal and spatial service coverage," *IEEE Trans. Veh. Technol.*, vol. 71, no. 5, pp. 4596–4606, May 2022.

[3] A. Moubayed, A. Shami, P. Heidari, A. Larabi, and R. Brunner, "Edge-enabled V2X service placement for intelligent transportation systems," *IEEE Trans. Mobile Comput.*, vol. 20, no. 4, pp. 1380–1392, Apr. 2021.

[4] T. A. Edwan, A. Tahat, H. Yanikomeroglu, and J. Crowcroft, "An analysis of a stochastic ON-OFF queueing mobility model for software-defined vehicle networks," *IEEE Trans. Mobile Comput.*, vol. 21, no. 5, pp. 1552–1565, May 2022.

[5] X. Li, S. Chen, Y. Zhou, J. Chen, and G. Feng, "Intelligent service migration based on hidden state inference for mobile edge computing," *IEEE Trans. Cogn. Commun. Netw.*, vol. 8, no. 1, pp. 380–393, Mar. 2022.

[6] J. Xu, X. Ma, A. Zhou, Q. Duan, and S. Wang, "Path selection for seamless service migration in vehicular edge computing," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 9040–9049, Sep. 2020.

[7] W. Lu, X. Meng, and G. Guo, "Fast service migration method based on virtual machine technology for MEC," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4344–4354, Jun. 2019.

[8] F. Zhang, G. Liu, X. Fu, and R. Yahyapour, "A survey on virtual machine migration: Challenges, techniques, and open issues," *IEEE Commun. Surv. Tut.*, vol. 20, no. 2, pp. 1206–1243, Second Quarter 2018.

[9] C. Zhou, C. Feng, and Y. Wang, "Spatial-temporal energy management of base stations in cellular networks," *IEEE Internet Things J.*, vol. 9, no. 13, pp. 10588–10599, Jul. 2022.

[10] T. Wang, Y. Shen, L. Gao, Y. Jiang, T. Ma, and X. Zhu, "Energy consumption minimization with throughput heterogeneity in wireless-powered body area networks," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3369–3383, Mar. 2021.

[11] M. R. Anwar, S. Wang, M. F. Akram, S. Raza, and S. Mahmood, "5G-enabled MEC: A distributed traffic steering for seamless service migration of Internet of Vehicles," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 648–661, Jan. 2022.

[12] J. Li, X. Shen, L. Chen, J. Ou, L. Wosinska, and J. Chen, "Delay-aware bandwidth slicing for service migration in mobile backhaul networks," *J. Opt. Commun. Netw.*, vol. 11, no. 4, pp. B1–B9, 2019.

[13] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung, "Dynamic service migration in mobile edge computing based on Markov decision process," *IEEE/ACM Trans. Netw.*, vol. 27, no. 3, pp. 1272–1288, Jun. 2019.

[14] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2333–2345, Oct. 2018.

[15] A. Machen, S. Wang, K. K. Leung, B. J. Ko, and T. Salonidis, "Live service migration in mobile edge clouds," *IEEE Wireless Commun.*, vol. 25, no. 1, pp. 140–147, Feb. 2018.

[16] C. Zhu, G. Pastor, Y. Xiao, Y. Li, and A. Ylae-Jaeaeski, "Fog following me: Latency and quality balanced task allocation in vehicular fog computing," in *Proc.IEEE 15th Annu. Int. Conf. Sens. Commun. Netw.*, 2018, pp. 1–9.

[17] A. Aissioui, A. Ksentini, A. M. Gueroui, and T. Taleb, "On enabling 5G automotive systems using follow me edge-cloud concept," *IEEE Trans. Veh. Technol.*, vol. 67, no. 6, pp. 5302–5316, Jun. 2018.

[18] M. Chowdhury, E. Steinbach, W. Kellerer, and M. Maier, "Context-aware task migration for HART-centric collaboration over FiWi based tactile internet infrastructures," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 6, pp. 1231–1246, Jun. 2018.

[19] S. Jeong, N. Van Tu, J.-H. Yoo, and J. W.-K. Hong, "Proactive live migration for virtual network functions using machine learning," in *Proc. IEEE 17th Int. Conf. Netw. Serv. Manage.*, 2021, pp. 335–339.

[20] L. Yu et al., "STEP: A spatio-temporal fine-granular user traffic prediction system for cellular networks," *IEEE Trans. Mobile Comput.*, vol. 20, no. 12, pp. 3453–3466, Dec. 2021.

[21] X. Zhou, S. Ge, T. Qiu, K. Li, and M. Atiquzzaman, "Energy-efficient service migration for multi-user heterogeneous dense cellular networks," *IEEE Trans. Mobile Comput.*, vol. 22, no. 2, pp. 890–905, Feb. 2023.
[22] U. Fattore, M. Liebsch, B. Brik, and A. Ksentini, "AutoMEC: LSTM-based user mobility prediction for service management in distributed MEC resources," in *Proc. 23rd Int. ACM Conf. Model. Anal. Simul. Wireless Mobile Syst.*, 2020, pp. 155–159.
[23] Q. Yuan, J. Li, H. Zhou, T. Lin, G. Luo, and X. Shen, "A joint service migration and mobility optimization approach for vehicular edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 9041–9052, Aug. 2020.
[24] I. Labriji et al., "Mobility aware and dynamic migration of MEC services for the Internet of Vehicles," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 1, pp. 570–584, Mar. 2021.
[25] S. Abdulah et al., "Accelerating geostatistical modeling and prediction with mixed-precision computations: A high-productivity approach with PaRSEC," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 4, pp. 964–976, Apr. 2022.
[26] Z. Han, H. Tan, G. Chen, R. Wang, Y. Chen, and F. C. Lau, "Dynamic virtual machine management via approximate Markov decision process," in *Proc. IEEE 35th Annu. Int. Conf. Comput. Commun.*, 2016, pp. 1–9.
[27] A. Filali, Z. Mlika, S. Cherkaoui, and A. Kobbane, "Preemptive SDN load balancing with machine learning for delay sensitive applications," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 15947–15963, Dec. 2020.
[28] W. Liu and Y. Shoji, "DeepVM: RNN-based vehicle mobility prediction to support intelligent vehicle applications," *IEEE Trans. Ind. Inform.*, vol. 16, no. 6, pp. 3997–4006, Jun. 2020.
[29] Y. Sun, S. Qin, G. Feng, L. Zhang, and M. A. Imran, "Service provisioning framework for RAN slicing: User admissibility, slice association and bandwidth allocation," *IEEE Trans. Mobile Comput.*, vol. 20, no. 12, pp. 3409–3422, Dec. 2021.
[30] O. Alamu, A. Gbenga-Ilori, M. Adelabu, A. Imoize, and O. Ladipo, "Energy efficiency techniques in ultra-dense wireless heterogeneous networks: An overview and outlook," *Eng. Sci. Technol. Int. J.*, vol. 23, no. 6, pp. 1308–1326, 2020.
[31] J. Fan, J. Bai, Z. Li, A. Ortiz-Bobea, and C. P. Gomes, "A GNN-RNN approach for harnessing geospatial and temporal information: Application to crop yield prediction," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 11873–11881.
[32] L. Wang, A. Adiga, J. Chen, A. Sadilek, S. Venkatramanan, and M. Marathe, "CausalGNN: Causal-based graph neural networks for spatio-temporal epidemic forecasting," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 12191–12199.
[33] A. Shewalkar, "Performance evaluation of deep neural networks applied to speech recognition: RNN, LSTM and GRU," *J. Artif. Intell. Soft Comput. Res.*, vol. 9, no. 4, pp. 235–245, 2019.
[34] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, *arXiv:1412.3555*.
[35] M. Welling and T. N. Kipf, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2016, pp. 1–14.
[36] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," in *Proc. IEEE 29th Conf. Decis. Control*, 1990, pp. 2130–2132.
[37] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synth. Lectures Commun. Netw.*, vol. 3, no. 1, pp. 1–211, 2010.
[38] L. Bieker, D. Krajzewicz, A. Morra, C. Michelacci, and F. Cartolano, "Traffic simulation for all: A real world traffic scenario from the city of bologna," in *Modeling Mobility with Open Data*. Berlin, Germany: Springer, 2015, pp. 47–60.
[39] L. Codeca, R. Frank, S. Faye, and T. Engel, "Luxembourg SUMO traffic (LuST) scenario: Traffic demand evaluation," *IEEE Intell. Transp. Syst. Mag.*, vol. 9, no. 2, pp. 52–63, Summer 2017.
[40] E. J. Oughton, Z. Frias, S. van der Gaast, and R. van derBerg, "Assessing the capacity, coverage and cost of 5G infrastructure strategies: Analysis of The Netherlands," *Telematics Inform.*, vol. 37, pp. 50–69, 2019.
[41] M. M. Ahamed and S. Faruque, "5G network coverage planning and analysis of the deployment challenges," *Sensors*, vol. 21, no. 19, 2021, Art. no. 6608.
[42] Y. Li, Y. Yin, X. Chen, J. Wan, G. Jia, and K. Sha, "A secure dynamic mix zone pseudonym changing scheme based on traffic context prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 9492–9505, Jul. 2022.
[43] D. Brain and G. I. Webb, "On the effect of data set size on bias and variance in classification learning," in *Proc. Fourth Australian Knowl. Acquisition Workshop*, 1999, pp. 117–128.
[44] M. Brambilla, D. Pardo, and M. Nicoli, "Location-assisted subspace-based beam alignment in LOS/NLOS mm-wave V2X communications," in *Proc. IEEE Int. Conf. Commun.*, 2020, pp. 1–6.
[45] X. Ma and K. S. Trivedi, "SINR-based analysis of IEEE 802.11p/bd broadcast VANETs for safety services," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 3, pp. 2672–2686, Sep. 2021.
[46] M. Yan et al., "Modeling the total energy consumption of mobile network services and applications," *Energies*, vol. 12, no. 1, 2019, Art. no. 184.

Chenglong Wang (Student Member, IEEE) received the BE degree from the School of Computer Science and Engineering, Central South University, Changsha, China, in 2018. He is currently working toward the PhD degree with Central South University. He is currently a visiting PhD student with the Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada. His current research interests include mobility prediction for connected vehicles and resource management for edge networks.

Jun Peng (Senior Member, IEEE) received the BS degree from Xiangtan University, Xiangtan, China, in 1987, the MSc degree from the National University of Defense Technology, Changsha, China, in 1990, and the PhD degree in control science and control engineering from Central South University, Changsha, in 2005. She is currently a professor with the School of Computer Science and Engineering, Central South University. In April 1990, she joined the staff of Central South University. From 2006 to 2007, she was with the School of Electrical and Computer Science, University of Central Florida, Orlando, FL, USA, as a visiting scholar. Her research interests include cooperative control and cloud computing and wireless communications.

Lin Cai (Fellow, IEEE) received the MASc and PhD degrees (awarded Outstanding Achievement in Graduate Studies) in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2002 and 2005, respectively. Since 2005, she has been with the Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada, and she is currently a professor. Her research interests include communications and networking, with a focus on network protocol and architecture design supporting emerging multimedia traffic and the Internet of Things. She is an NSERC E.W.R. Steacie Memorial Fellow. In 2020, she was elected as a member of the Royal Society of Canada's College of New Scholars, Artists, and Scientists. She was also elected as a 2020 Star in Computer Networking and Communications by N2Women. She was the recipient of NSERC Discovery Accelerator Supplement Grants in 2010 and 2015, respectively, and the best paper awards of IEEE ICC 2008 and IEEE WCNC 2011. She has co-founded and chaired the IEEE Victoria Section Vehicular Technology and Communications Joint Societies Chapter. Since 2019, she has been elected to serve the *IEEE Vehicular Technology Society Board of Governors*. She was an area editor of *IEEE Transactions on Vehicular Technology*, a Member of the Steering Committee of *IEEE Transactions on Big Data* and *IEEE Transactions on Cloud Computing*, an associate editor for the *IEEE Internet of Things Journal*, *IEEE Transactions on Wireless Communications*, *IEEE Transactions on Vehicular Technology*, *IEEE Transactions on Communications*, *Eurasip Journal on Wireless Communications and Networking*, *International Journal of Sensor Networks*, and *Journal of Communications and Networks*, and the *Distinguished Lecturer of the IEEE VTS Society*. She was a TPC Co-Chair for IEEE VTC2020-Fall and a TPC Symposium Co-Chair for IEEE Globecom'10 and Globecom'13. She is a registered Professional Engineer in British Columbia, Canada.

Hui Peng received the BEng degree in school of automation from Beijing Information Science and Technology University, Beijing, China, in 2013, and the PhD degree in school of automation from Beijing Institute of Technology, Beijing, in 2019. From 2019 to 2021, he was a postdoctoral fellow of engineering with the department of automation, college of intelligence science and technology, National University of Defense Technology, Changsha, China. Since September 2021, he has been with the School of Computer Science and Engineering, Central South University, where he is currently a Lecturer. His research interests include edge computing, artificial intelligence, and robot systems.

Weirong Liu (Member, IEEE) received the BE degree in computer software engineering and the ME degree in computer application technology from the Central South University, Changsha, China, in 1998 and 2003, respectively, and the PhD degree in control theory and control engineering from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2007. Since 2008, he has been a faculty member with the School of Information Science and Engineering, Central South University, where he is currently a professor. His research interests include cooperative control, energy storage management, reinforcement learning, neural networks, wireless sensor networks, network protocol, and microgrids.

Xin Gu (Student Member, IEEE) received the BS degree in communication engineering from Central South University, Changsha, China, in 2015. He is currently working toward the PhD degree in the School of Automation, Central South University. She is currently a visiting PhD student with the Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada. Her research interests include cellular vehicle-to-everything (C-V2X), wireless resource management, and protocol design.

Zhiwu Huang (Member, IEEE) received the BS degree in industrial automation from Xiangtan University, Xiangtan, China, in 1987, the MS degree in industrial automation from the University of Science and Technology Beijing, Beijing, China, in 1989, and the PhD degree in control theory and control engineering from Central South University, Changsha, China, in 2006. He is currently a professor with the School of Automation, Central South University. In October 1994, he joined the staff of Central South University. From 2008 to 2009, he was with the School of Computer Science and Electronic Engineering, University of Essex, Colchester, U.K., as a visiting scholar. His research interests include fault diagnostic technique and cooperative control.