A Novel Prediction-Based Temporal Graph Routing Algorithm for Software-Defined Vehicular Networks

Liang Zhao, *Member, IEEE,* Zhuhui Li, Ahmed Al-Dubai, *Senior Member, IEEE,* Geyong Min, Jiajia Li, *Member, IEEE,* Ammar Hawbani and Albert Y. Zomaya, *Fellow, IEEE*

***Abstract*—Temporal information is critical for routing computation in the vehicular network. It plays a vital role in the vehicular network. Till now, most existing routing schemes in vehicular networks consider the networks as a sequence of static graphs. We need to find an appropriate method to process temporal information into routing computation. Thus, in this paper, we propose a routing algorithm based on the Hidden Markov Model (HMM) and temporal graph, namely, Prediction-Based Temporal Graph Routing Algorithm (PT-GROUT). This new algorithm considers the vehicular network as a temporal graph, in which each data transmission as an edge has its specific temporal information. To better capture the temporal information, we select Software-Defined Vehicular Network (SDVN) as our network architecture, which is a preferred architecture for processing the temporal graph regarding the vehicular network since all vehicle statuses can be easily managed. To compute the future routing path accurately and efficiently, the future temporal graph is predicted by applying HMM, in which we model the current vehicular network with dynamic programming and greedy strategies. With the temporal information and reasonable setting of HMM, PT-GROUT can better evaluate the vehicular network and discover the evolution of the internal structure of the network. The optimal routing path can be achieved more efficiently. The simulation results demonstrate that PT-GROUT can substantially improve the computation efficiency and reduce packet loss and delivery delay compared with its counterparts.**

***Index Terms*—vehicular ad hoc networks, routing, software-defined vehicular networks, hidden markov model, temporal graph.**

I. Introduction

Vehicular ad-hoc network (VANET) has been instrumental in the Intelligent Transportation System (ITS) [1]. A great deal of studies has been conducted on VANET technologies in recent years, whereas traditional distributed-fashion ad-hoc protocols have no longer satisfied the boosting demand nowadays. Repeated and redundant computation in a distributed manner dragged down the evolution of VANET. This redundancy mainly occurs on two reasons. First, due to the limited vision brought by the distributed fashion of VANET, each vehicle can sense the status of its neighbor easily, whereas collecting extra information beyond one hop will create a huge overhead. The computed routing path based on the limited information could not be optimal globally. Second, the computational ability of a vehicle is very limited, which constrains the feasibility of analyzing and learning the routing patterns from the big vehicular data. Therefore, as a novel networking paradigm, Software-Defined Vehicular Network (SDVN) makes up for the shortcomings caused by the current architecture of vehicular communication [2]. SDVN is an emerging architecture combining the Software-Defined Network (SDN) and VANET that allows for centralized management and distributed policy control [3]. The separation of the data plane and the control plane is the core idea of SDVNs. The data plane refers to vehicles and roadside units equipped with sensors and transmitters. The logically centralized controller sits at the control plane to programmable functions. There could be multiple controllers cooperating to realize all functions to support the data plane in real-life. The whole control plane coordinates the computing tasks among multiple controllers through reasonable architecture settings and edge computing technologies [4], [5]. With reducing the computational burden greatly, it ensures efficient and accurate support for the data plane. Since the vehicle's status data is collected from the data plane periodically by the controller, the controller can sense the vehicular network globally [6], [7]. And, based on this collected knowledge, the controller can efficiently compute the globally optimal solution for the entire network. Moreover, the nodes on the data plane are required to focus on forwarding packets and updating vehicle status information only, instead of spending time and resources on routing computation. The separation of data plane and control plane will bind network elements on the data plane closer, allowing for more flexibility and scalability of the vehicular network itself.

Similar to other networking, routing is also the basis for vehicular data transmission by finding a vehicle sequence (i.e., routing path) from the requester vehicle to the destination vehicle, while later packets (i.e., data messages) will be transmitted along this sequence. The quality of routing affects the QoS of data transmission directly. In VANETs, every vehicle takes responsibility for routing computation and maintenance tasks. In such a scheme, each vehicle selects the next-hop vehicle based on the surrounding environment consisting of a limited number of neighboring vehicles. However, the statuses of these vehicles are not enough to reflect the characteristics of the entire vehicular network for the optimal routing computation.

Liang Zhao, Jiajia Li, and Zhuhui Li are with School of Computer Science, Shenyang Aerospace University, Shenyang, China. E-mail: {lzhao, lijiajia}@sau.edu.cn, zhuhui.li7217@gmail.com.

Ahmed Al-Dubai is with School of Computing; Edinburgh Napier University, UK. E-mail: a.al-dubai@napier.ac.uk

Geyong Min is with the Department of Computer Science, University of Exeter, UK. E-mail: g.min@exeter.ac.uk

Ammar Hawbani is School of Computer Science and Technology, University of Science and Technology of China, Hefei, China. E-mail: anmande@ustc.edu.cn

Albert Zomaya is with University of Sydney, Australia; E-mail: albert.zomaya@sydney.edu.au.

Manuscript received XX X, XX; revised XX XX, XX.

SDVN could be the right candidate for solving these issues. The controller can always get the global view of the entire vehicular network (i.e., the topological information of the network) based on the information it collected from the data plane. This global view of the controller can guarantee the quality and successful delivery ratio of routing for each vehicle in the vehicular network.

In fact, the quality of vehicular networking depends on the routing algorithm. Once the routing algorithm is able to capture the properties of the vehicular network better, data transmission can achieve higher QoS provisioning. On the other hand, as the controller needs to cope with the routing request for the entire vehicular network, the computation efficiency could be a potential challenge. If some emergency incidents (e.g., car crash) happen, the controller needs to process a large amount of routing requests in a concise period to ensure the fast spreading of safety-related information. If the controller cannot handle these complex computation tasks efficiently, the increased delay of data packets can cause serious consequences [8]. These are several challenges that the routing algorithm must face in SDVNs. Among them, there are two vital factors, *routing performance* and *routing computation efficiency*, that will directly affect the performance of data transmission. Thus, in this work, we would like to design a routing algorithm to guarantee the quality and efficiency of routing computation simultaneously.

The temporal graph can simultaneously meet both demands mentioned above [9], [10]. First, existing SDVN routing schemes consider the vehicular network as a sequence of static graphs at different timestamps [11], [12]. Each routing request is processed on an independent static graph in the controller. With the time evolution, the correlation between static graphs at different timestamps contains a large amount of temporal-related network information. We call them *temporal information*. Specific to routing, the temporal information contains the starting time and duration of data transmission. This is critical for capturing the temporal-related properties of the vehicular network, as routing is time-dependent [13]. Adopting the temporal graph instead of the static graph to represent the entire VANET can be positive to capturing network characteristics and attributes. On the other hand, routing in the SDVN is basically an optimal path problem. The efficiency of optimal path computation is even better than the most efficient optimal path algorithm of the static graph [9], [14]. Overall, the introduction of temporal graph can solve the quality and the efficiency of routing computation at the same time. However, involving temporal graph poses a new challenge. The temporal graph is commonly used to record network information, which has happened in the past. For example, the determined network information, like flight information, can be recorded in the form of the temporal graph. Conversely, the goal of routing is to plan the uncertain routing transmission path in the future. Hence, we need to utilize *prediction* to apply an efficient temporal graph method for the computation of routing (flow-tables). Meanwhile, it is difficult for routing algorithms to pursue the tradeoff between the quality of the routing path and computation efficiency [12]. The consideration of one single optimization method mentioned above is not adequate. We have to find an appropriate method to smoothly integrate temporal graph into routing computation in the vehicular network. First, we design a prediction algorithm to capture future network status and temporal information as much as possible to form the temporal graph representing the future vehicular network. Then, we employ the efficient optimal path algorithm of temporal graph to process the graph to obtain the optimal routing path. We have to make sure every step extremely efficient to avoid additional computation waste and network overhead for the entire network architecture.

Thus, in this paper, we propose a novel routing algorithm, namely, Prediction-Based Temporal Graph Routing Algorithm (PT-GROUT), for SDVNs. At first, we utilize the Hidden Markov Model (HMM) to predict future routing information, as the input of the optimal routing algorithm [15]. Besides the traditional properties (e.g., source and destination), the temporal properties are also involved. In HMM, the state is the intersection, and the observation is the vehicles around the corresponding intersection. Historical routing information and position information are utilized to predict the future temporal graph. There are two requirements for this temporal graph, (1) this graph should include all routing possibilities, and (2) the size of this graph should be limited to guarantee the prediction and computation efficiency. Finally, an efficient optimal routing algorithm is proposed to handle the predicted temporal graph for the optimal routing path. This can improve the efficiency performance of the routing algorithm significantly at a high level. The major contributions of this paper can be summarized as follows.

- Under a novel model combined with greedy and dynamic programming strategies, an HMM is constructed based on the current vehicular network efficiently. All parameters of this model are adaptively adjusted according to the source vehicle, destination vehicle, and the current status of the vehicular network.
- A novel prediction strategy is proposed to predict a temporal graph representing the possible future routing based on the constructed HMM. All possibilities of routing can be included in this graph, and the size of this temporal graph is limited by our algorithm to improve the routing path computation efficiency. At this point, we involve the concept of temporal graph for routing. Due to the temporal information the graph brings with, the routing path computed based on this temporal information will be efficient and realistic.
- An efficient optimal routing algorithm for the temporal graph is utilized by applying the properties of temporal graphs. Within linear time complexity, the single-source shortest path can be achieved.

The rest of the paper is organized as follows. Section II introduces the related work about VANET and SDVN. Section III presents the network model of our algorithm and problem formulation in detail. Our proposed algorithm is described in detail in Section IV. In Section V, simulation results are presented and analyzed from different aspects. Finally, we conclude this paper in Section VI. The comparison with our previous work and the time complexity analysis is presented

in the supplementary material.

II. Related Work

Many routing algorithms have been proposed for VANETs. From architecture to method, routing algorithms in VANETs are gradually evolving to adapt to the increasingly complex topology of the vehicle network. In this section, we briefly review related work on routing algorithms in the following four aspects, traditional routing algorithms, predicted-based routing algorithms, routing algorithms of SDVNs, and temporal-related routing applications in VANET.

In terms of the information the algorithm demands, traditional routing algorithms proposed for ad-hoc network can be classified into four different categories, topology-based routing, position-based routing, map-based routing, path-based routing [16]. In the distributed routing methods such as GPSR [17] and AODV [18], all vehicles collect the status information about surrounding vehicles by sending beacons. Each related vehicle calculates the next-hop vehicle or flow table in this context, then forwards the data message. This may simplify the algorithm. However, the limited resource and vision of vehicle nodes could lead to the deterioration of routing quality.

Many researchers integrate the prediction algorithms into their routing schemes to pursue a better quality of routing path. Namboodiri et al. propose a prediction-based routing (PBR) algorithm in [19]. This algorithm applies the predicted route lifetime to create new routes before existing ones fail preemptively. In [20], Yao et al. propose a V2X routing scheme PRHMM by adopting HMM in the prediction task. They apply HMM to predict the future position and the future relationship with the destination. Then, they build two metrics calculated based on the predicted data, delivery probability, and end-to-end delay. Each vehicle uses these two metrics to select the next-hop vehicle or RSUs.

Even with the predictive mechanisms, the finiteness vision of individual vehicles still limits the quality and computational efficiency of routing. Since the architecture of SDVN was proposed, many routing-related studies have been conducted in order to implement efficient routing computation in such architecture. Rayeni et al. [11] propose an optimal resource utilization routing scheme (ORUR), which involves existing routing paths to relay data. Load balancing has also been considered in their proposed scheme to minimize the congestion. However, the routing process combines Dijkstra and parallel computing with a high computational complexity which imposes a heavy burden on the controller. Gao et al. [12] propose a hierarchical routing scheme considering load balancing in SDVNs. They divide the routing processing into three parts, grid selection, segment selection, and vehicle selection. Each part takes several related properties into consideration. After all, a series of relay nodes can be achieved as a routing path for the requester vehicle efficiently. However, at each part, this scheme fails to avoid local optimum, which results in degradation of the quality of computation routing. Yan et al. [21] propose a link available time prediction-based backup caching and routing (LBR) scheme, which is suitable for high-Speed Vehicular Networks, especially the high-speed railway scenario. In such a scenario, the computation efficiency and the prediction accuracy become more important. In [22], Zhao et. al propose a hierarchical greedy routing scheme considering link stability. The fuzzy logic trained by reinforcement learning is utilized to assist the routing decision. Even it achieved a significant improvement in performance, the complexity of this algorithm will rise exponentially with the increase of the network size. More detailed comparisons of SDVN routing algorithms are presented in Table I.

On the other hand, temporal information has not been considered in all these SDVN routing algorithms. They all treat the network as a static graph and apply the traditional static graph algorithm to the vehicular network. However, the network is always dynamic evolving, modeling the network at a certain future timestamp and computing on it are unable to evaluate the network. Observed from the following temporal information related work, temporal information is very helpful for analyzing network conditions. At the same time, in the above work, the static graph-based traditional algorithm cannot always find the tradeoff between computation quality and efficiency. Some of them cannot achieve the optimal result stably, while others improve the computation efficiency sacrificing additional computation resource. It is difficult for existing routing algorithms to pursue the tradeoff between the quality of routing path and computation efficiency [12]. The optimal routing algorithm considering the temporal graph could be the answer since it can compute the optimal routing under the linear time and space complexity. The excellent performance of temporal graph in the computation efficiency and analyzing network drives us to introduce it into the vehicle network routing calculation.

Meanwhile, the importance of temporal information has not been considered seriously. As for the temporal information in VANETs, researchers rarely use it for routing computation. In [23], Scellato et al. evaluate the temporal robustness of mobile networks. Their work is the first attempt to define the concept of temporal network robustness, where they describe a measure of network robustness for time varying networks. Qiao et al. add the characteristics of temporal structural to VANETs, including temporal network efficiency and temporal closeness centrality [24]. At last, they measure several metrics based on a Taxi GPS dataset. Based on the above work, we can see that temporal information can help us better analyze and predict the possible future conditions and trends. However, there are no existing studies introducing temporal information into the routing computation. To the best of our knowledge, this work is the first to integrate temporal information into routing computation.

Thus, we decide to adopt the network architecture of SDVNs. Based on the advantage of the global view, we construct the HMM to integrate the prediction into our algorithm. The temporal graph representing the future vehicular network based on HMM is constructed. At last, we utilize an optimal routing algorithm under the temporal graph. The efficient temporal-graph-based optimal routing algorithm can improve the performance significantly at a high level.

TABLE I
CHARACTERISTICS OF ROUTING SCHEMES IN SDVN (NOTES: N/A-NOT AVAILABLE IN THE RELATED ARTICLE)

	Focus	Fundamental Algorithm	Complexity	Scalability	Advantage	Application Scenario
PT-GROUT	Temporal information, Computation efficiency	Temporal optimal path algorithm	Low	Medium	Efficient algorithm, Stable QoS	Urban
ORUR [11]	Load balance	Dijkstra	High	High	QoS improvement in terms of channel busy ratio	Dense urban
LBR [21]	Link available time prediction	N/A	Low	Medium	High and reliable delivery ratio	High-speed railway
GLS [22]	Predictive delivery rate Link stability	Greedy, Fuzzy logic	Medium	Low	High delivery ratio and forwarding efficiency.	Dense urban
HRLB [12]	Load balance	Greedy	Low	Low	High delivery ratio, throughput, average delay	Urban

TABLE II
NOTATIONS

Notation	Description
$REI(u, v, t, d)$	The Routing Edge Information
dis_{ij}	The distance between intersection i and intersection j
s_{ij}	The segment originated from intersection i to intersection j
$den(s_{ij})$	The vehicle density of the segment s_{ij}
$V(s_{ij})$	The number of vehicles on the segment s_{ij}
$l(s_{ij})$	The length of the segment s_{ij}
$rdex_{ij}$	The historical intersection routing index from intersection i to intersection j
$tdex_{ij}$	The transition index from intersection i to intersection j
t_h	The time value at time h
$r_{ij}^{t_h}$	Routing valid value from intersection i to intersection j at time h
α	The weight coefficient of the dynamic programming part
k	The weight coefficient of vehicle density
$dis_{v_x,ij}$	The distance before vehicle x reaches the boundary between intersection i and intersection j
q_x	The qualification index of the vehicle x

III. NETWORK MODEL AND PROBLEM FORMULATION

In this section, we describe the network model to present our proposed algorithm better. The problem formulation of this work will be presented in Section II(B). The notations of this work are listed in Table II.

A. Network Model

Fig. 1 shows the two bottom-layer components of SDVN, control plane and data plane. Here, the data plane includes base stations (BSs) and vehicles. Since SDVN needs stable communication between the data plane and the control plane, we adopt BSs as relay nodes to guarantee reliable and stable data transmission towards the control plane. There are two types of communications, vehicle to vehicle (V2V) communication, and vehicle to BS (V2B) communication. In this work, we use LTE-V (Device-to-Device) for V2V, and 5G or LTE for V2B [25].

All vehicles are required to send beacon messages to the control plane via BSs at a certain time interval. In this way, the control plane can get the latest global view of the entire vehicular network immediately. Each beacon message contains the vehicle's position, velocity, and acceleration. Hence, the beacon message can present the specific status of the vehicle in vehicular networks. With beacon messages collected from the data plane, the control plane can construct a global view of this vehicular network. To handle the highly mobile nature of the vehicles, we have to ensure that the control plane always has a global view of this network, even the beacon messages are lost during the transmission. Thus, the controller can predict and construct the status of the corresponding vehicle according to the latest received messages stored in the controller. Once we need to transmit data, the source vehicle sends a routing request to the control plane through V2B communication. Several centralized controllers of SDVN will cooperate and serve as the primary computing device of the control plane. When the routing request is received, the control plane computes the optimal routing path with its efficient algorithm and sends it back to the vehicles on the computed routing path through V2B communication. The flow tables of these corresponding vehicles will be updated. At last, the vehicles on the optimal routing path forward the packet according to their flow tables. When vehicles condition is too parse, part of BSs will be considered as nodes to assist vehicles in transmitting packets as relay nodes. Meanwhile, as the vehicle density increases, the number of BSs which play the roles of relay nodes will drop, and more BSs will only be responsible for routing request transmission. In this method, the tradeoff between data transmission success ratio and the transmission load of BSs can be achieved.

During data forwarding, if one vehicle cannot receive the ACK (ACKnowledgement) message from the corresponding next-hop vehicle after a certain time interval. This part of data transmission will be considered as a failed data forwarding.

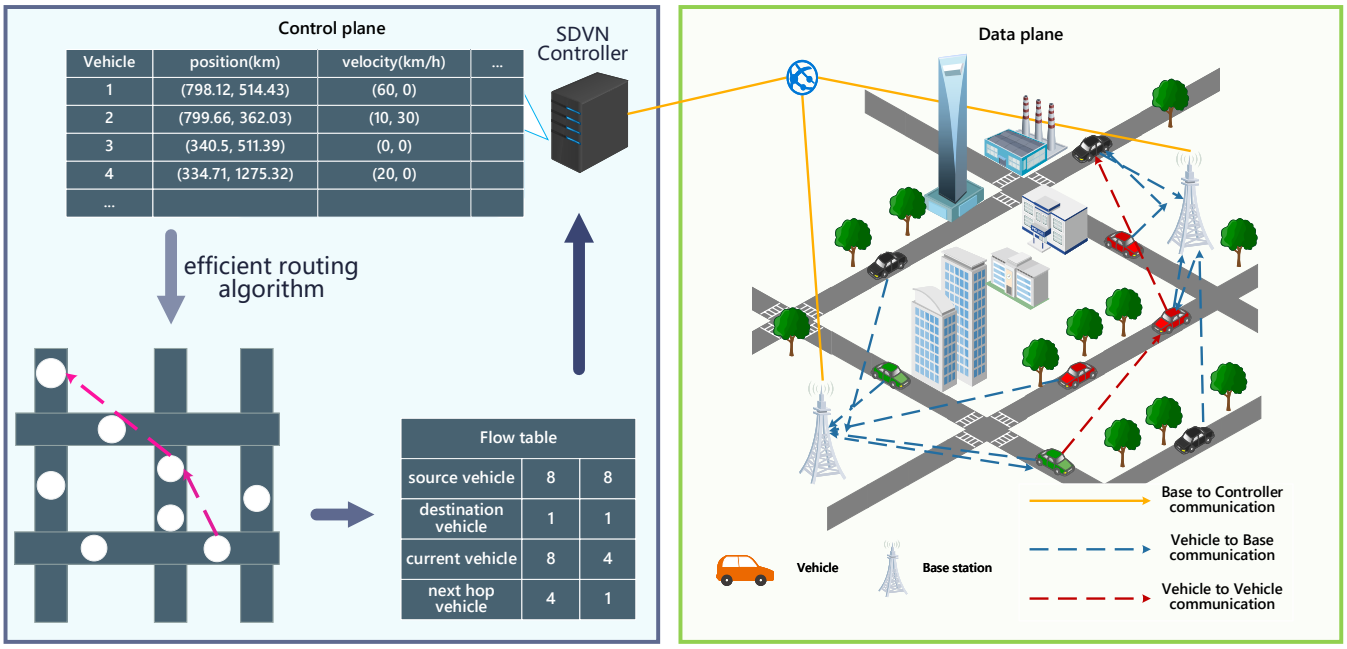


Fig. 1. A typical urban scenario of SDVNs (one vehicle requesting for transmitting traffic information to another vehicle).

In case of failed data forwarding, the failure vehicle will send the error report message to the controller for re-calculating the routing path. According to the re-calculated routing path, the relevant vehicles which failed in the data transmission will forward the data messages along the new path. If the data message is received by the destination node successfully this time, the repair process is then accomplished. If the same routing errors occur three times consecutively, this routing discovery will be regarded as a failure. At the same time, all related routing information of this failed routing will be deleted in both control plane and data plane to ensure the reliability and real-time of the routing information. If the source vehicle cannot reach the controller, it will use the traditional GPSR to forward the packets until the packets are received by the vehicle that can sense the control plane. Then, this vehicle will employ the PT-GROUT algorithm to continue data transmission [17].

The channel model employed in this paper is shown as follows, First, when vehicle s sends data packets to vehicle d, the interference of vehicle d at t will be calculated based on the path loss model in [26], [27],

$$i_{s-d,t} = N + \sum_{i=1,i\neq s}^{n_{nei-d,t}} \xi\left(d_{i-d,t}\right) = N + \sum_{i=1,i\neq s}^{n_{nei-d,t}} \frac{G_i G_d \lambda^2 P_i}{(4\pi)^2 d_{i-d,t}^{a}} \tag{1}$$

where $n_{nei-d,t}$ is all neighbor vehicles of d at t; N is the additive white Gaussian noise (AWGN); P_i is the transmission power; G_i and G_d are the antenna gains of sender and receiver; λ is the wavelength; $d_{i-d,t}$ denotes the Euclidean distance between the receiver vehicle d and its interference vehicle i at t. α is the path loss exponent and $2 \leq \alpha \leq 5$ depends on the geometry of propagation environment [28], We set $\alpha = 4$ since it represents the urban area cellular radio. For simplicity, let $G = \frac{G_i G_d \lambda^2}{(4\pi)^2}$, then (1) can be rewritten as:

$$i_{s-d,t} = N + G \sum_{i=1,i\neq s}^{n_{nei-d,t}} \frac{P_i}{d_{i-d,t}^{a}} \tag{2}$$

Based on the interference calculated by (2), the received signal to interference and noise ratio (SINR) between vehicle s and vehicle d at time t is denoted by $SINR_{s,d}^{t}$ (3),

$$SINR_{s-d,t} = \frac{\frac{GP_i}{d_{s-d,t}^{a}}}{N + G\sum_{i=1,i\neq s}^{n_{nei-d,t}} \frac{P_i}{d_{i-d,t}^{a}}} = \frac{d_{s-d,t}^{-a}}{\frac{N}{GP_i} + \sum_{i=1,i\neq s}^{n_{nei-d,t}} d_{i-d,t}^{-a}} \tag{3}$$

If the vehicle d can receive the data packet that transmitted from vehicle s successfully, $SINR_{s-d,t}$ should satisfy the constraint as follows:

$$SINR_{s-d,t} = \frac{d_{s-d,t}^{-a}}{\frac{N}{GP_i} + \sum_{i=1,i\neq s}^{n_{nei-d,t}} d_{i-d,t}^{-a}} \geq \beta \tag{4}$$

β is the receiving threshold which can guarantee successful data packet decoding at the receiver. And we set β as 0 dB.

B. Problem Formulation

Our goal is to transmit a given number of packets from the source vehicle src to the destination node des successfully. Besides, the time cost of computation to obtain a routing path $C(S_i)$ should be minimized. Meanwhile, the quality of routing path needs to be guaranteed to make sure the successful delivery of data packets. In terms of the quality of routing, we aim to improve the delivery ratio and minimize the average end-to-end delay and the delivery delay jitter. To find the optimal solution for our routing problem, the efficiency and quality of routing computation are combined to evaluate the performance of the routing algorithms. First, we will give

specific notations of the relevant metrics and concepts as follows.

1) Notation: First, our optimized variable is the routing path $S_i(v_0, v_1, \ldots, v_n)$ for each packet pkt_i. It is combined with n vehicles. Here, v_0 and v_n represent the source vehicle src and destination vehicle des, respectively. After obtaining results (i.e., routing path), the vehicles of this path will forward the packet i by order.

Here, we use $C(S_i)$ to represent the time cost for computing the routing path S_i for the packet pkt_i in the control plane. This metric depends on the time complexity of the routing algorithm. It is worth stating that the repeated computation time for fixing the failed routing is also included. Thus, this is partly related to the quality of the routing path. If the forwarding process fails, the control plane needs to spend redundant time to compute for the same routing request again.

The other metric is the quality of the routing path. Three sub-metrics are utilized to evaluate this metric comprehensively: the delivery ratio r, the average delivery delay T, and the delivery delay jitter j. We consider there are packet set pkt_set_m that successfully arrived at their specific destination nodes in the entire packet set pkt_set_n. Then, the delivery ratio is defined as:

$$r = \frac{sizeof(pkt\_set_m)}{sizeof(pkt\_set_n)} \tag{5}$$

The delivery delay T is the average value of the delivery delay of successful packet transmissions in pkt_set_m. Only the delivery delay of the successful packet transmission can be taken into consideration, which can be formulated as:

$$T = \frac{\sum T(S_i)}{sizeof(pkt\_set_m)}, \qquad forall S_i, i \in pkt\_set_m \tag{6}$$

The delivery delay of the packet i: $T(S_i)$ is more complicated. It is combined with four parts:

$$T(S_i) = t_{veh\_con} + C(S_i) + t_{packet}(S_i) + t_{ef} \tag{7}$$

where the vehicle-controller communication time t_{veh_con} is the time cost of the source vehicle spending on communicating with the control plane. It includes the routing request sending time and the routing path receiving time:

$$t_{veh\_con} = t_{vb} + t_{bc} \tag{8}$$

where t_{vb} is the time cost of the transmission from the vehicle to its nearest BS. Here, the source vehicle needs to find a suitable BS to assist it in communicating with the control plane. t_{bc} denotes the time cost of the transmission from the BS to the control plane. If the placement of BSs and controllers is reasonable, the difference of t_{veh_con} is small under the different centralized routing algorithms. When comparing the delivery delay, there will not be a measurable difference in terms of this part of the time cost. In contrast, this part of the time cost does not exist in the distributed algorithm as there is no centralized control plane to request the routing path. Hence, our proposed algorithm needs to perform better in other parts of delay, such as computation time and the packet delivery delay, to hold or even exceed the delay performance of the distributed routing algorithm. Moreover, the computation time in the control plane $C(S_i)$ is the time cost of computing the routing path in the controller, which is also known as processing delay in the control plane. $C(S_i)$ can also affect the quality of the routing at the same time. The time cost spending on the routing computation is also included in the delivery delay too. Hence, low routing computation efficiency will also increase the delay. As stated above, this metric is involved in both major evaluation metrics, efficiency, and routing quality. Thus, it is vital for the performance evaluation of routing algorithms.

The packet delivery delay $t_{packet}(S_i)$ is the main component of $T(S_i)$. It is also an important metric to evaluate the quality of the routing path. It is formulated as follows.

$$t_{packet}(S_i) = \sum_{k=0}^{n-1}(t_{trans|pkt_i}^{v_k,v_{k+1}} + t_{prop}^{v_k,v_{k+1}} + t_{proc}^{v_k,v_{k+1}} + t_q^{v_k,v_{k+1}})$$
$$\forall v_k \in S_i, \forall S_i, i \in pkt\_set_m \tag{9}$$

where $t_{trans|pkt_i}^{v_k,v_{k+1}}$ represents the transmission delay in sending a packet pkt_i within a single wireless hop from the vehicle v_k to the v_{k+1}. And $t_{prop}^{v_k,v_{k+1}}$ is the propagation delay. It depends mainly on the distance between two vehicles. We have used the standard equations to model these two metrics [29]. In addition, $t_{proc}^{v_k,v_{k+1}}$ and $t_q^{v_k,v_{k+1}}$ are processing delay and queuing delay, respectively. Both of them are highly dependent on the vehicle's current state , where the excessive load may even cause serious problems such as packet loss. These two metrics have all been formulated by the standard equations in [30].

From this point of view, three metrics, including the vehicle load, the number of hops, and the distance, have become the key metrics for evaluating the statuses of the vehicle and affecting the time cost of data transmission. If the statuses of vehicles on S_i can be well modeled and evaluated, $t_{packet}(S_i)$ will be significantly reduced. Even the packet loss will happen if the load, channel condition can be well evaluated. Conversely, if the statuses of vehicles cannot be objectively evaluated, $t_{packet}(S_i)$ will be greatly extended, seriously affecting the performance of the delivery delay $T(S_i)$, and increasing the possibility of packet forwarding failure. Thus, $T(S_i)$ can directly reflect the quality of the computed routing path. In PT-GROUT, we aim at minimizing the total distance of the routing path and stabilize the number of hops.

The error fixing time t_{ef} is also the component of the delivery delay. Sometimes, it will play a vital role. If the packet forwarding process fails according to the computed routing path S_i, the failed vehicle needs to restart a whole set of routing procedures, including routing request, computation, and the reply distribution. Once the computed routing path cannot guarantee successful delivery, the whole routing process will be repeated multiple times. A large amount of time cost and computation resources will be wasted due to this redundant computation. Thus, it is necessary to optimize the routing algorithm to ensure the delivery success ratio for improving the routing quality to minimize the time cost of this part. The combination of the time cost components will reflect the structural features of the vehicular network.

The last metric is the delivery delay jitter j. It is the variance of the delays of all successfully delivered packets in pkt_set_m.

This metric can reflect the stability of the quality of computed routing under different scenarios. It is formulated as:

$$j = \frac{\sum(T(S_i)-T)^2}{sizeof(pkt\_set_m)}, \qquad \forall S_i, i \in pkt\_set_m \tag{10}$$

2) Objective Function: Our objective is to minimize the time cost of the routing computation. Meanwhile, we also need to guarantee the quality of the computed routing path. Under the premise of ensuring the minimum computation time cost, the networking performance in terms of delivery ratio, delay, and jitter is supposed to be optimal to ensure high-quality data packet transmission. The quality of the computed routing path should be equal or close to the optimal one, which means the feasible routing path with the minimum delay. Thus, the objective function can be formulated as in the following (11)-(19).

$$\begin{aligned} & main \quad objective \\ \min \quad & \sum C(S_l), \quad l \in pkt\_set_n && (11) \\ & sub \quad objective \\ \min \quad & T && (12) \\ \max \quad & r && (13) \\ \min \quad & j && (14) \\ & \text{s.t.} \end{aligned}$$

$$\sum_{\substack{j=1 \\ ((v_i,v_j)\in S_h)}}^{n} link_{v_i,v_j}^{S_h} - \sum_{\substack{j=1 \\ ((v_i,v_j)\in S_h)}}^{n} link_{v_j,v_i}^{S_h} = \begin{cases} 1 & v_i = src \\ -1 & v_i = des \\ 0 & v_i \neq src, des \end{cases}, \forall S_h, h \in pkt\_set_m \tag{15}$$

$$link_{v_i,v_j}^{S_h} \geq 0, \quad (v_i,v_j) \in S_h, \quad \forall S_h, h \in pkt\_set_m \tag{16}$$

$$link_{v_i,v_j}^{S_h} + link_{v_j,v_i}^{S_h} \leq 1, \quad (v_i,v_j) \in S_h, \quad \forall S_h, h \in pkt\_set_m \tag{17}$$

$$d_{v_i,v_{i+1}} \leq d_{trans}^{max}, \quad v_i \in S_h, \forall S_h, h \in pkt\_set_m \tag{18}$$

$$size_h^{v_i,v_j} \leq cap_{v_j}, \quad (v_i,v_j) \in V, \quad \forall S_h, h \in pkt\_set_m \tag{19}$$

First, the objective of our routing algorithm is divided into two parts, the main objective, and the sub-objective. The main objective, which is to minimize the computation time cost of our routing algorithm, should be satisfied with the highest priority. Then, under the premise that the main objective is well satisfied, three sub-objectives related to the quality of routing should be met too. It is worth to mention that the implementation of main and sub-objective does not conflict. If we can extract the characteristics of the network well through a well-designed routing algorithm, reasonable pre-computation and data processing, it is possible to pursue the high efficiency and routing quality at the same time [31]. In the constraints, $link_{v_j,v_i}^{S_h}$ means the link state between vehicle v_i and vehicle v_j during the routing computation (15). It is a boolean value. Once v_i needs to transmit packets to v_j during the forwarding process of S_h, the $link_{v_j,v_i}^{S_h}$ will be true. Otherwise, it is false. (16) and (17) are based on the flow conservation concept which is responsible for routing computation for all packets. (17) is utilized to avoid the loops in the routing computation. In the next constraint (18), $dis_{v_i,v_{i+1}}$ means the distance between v_i and v_{i+1}. dis_{trans}^{max} denotes the maximum transmission distance between two communicating vehicles. We need to ensure the distance of each adjacent vehicle pair in S_h within this threshold. Otherwise, the data transmission based on this routing S_h is bounded to fail. The last constraint (19) is related to capacity. $size_h^{v_i,v_j}$ represents the size of packet h in which this packet is transmitted from v_i to v_j. cap_{v_j} means the remaining capacity of v_j. If the receive queue of the vehicle is full, this vehicle will not be taken into consideration during the routing computation until the receive queue of this vehicle becomes spare.

IV. Prediction-Based Temporal Graph Routing

In this section, we first introduce the reason that we adopt the HMM as our prediction model. Then, we present PT-GROUT in detail. In PT-GROUT, we set the intersections as the states and the vehicles around the intersections as the observations. The HMM representing the current network is constructed with dynamic programming and greedy strategies. After that, the intersection sequence, which is most likely to be the routing path generated base on this model. At last, we can construct the temporal graph by linking the vehicles on the intersection sequence. Since the edge in the temporal graph, which represents the routing information between vehicles, is different from the edge in the traditional concept. Therefore, we give the definition of the routing edge information as follows.

DEFINITION 1 (Routing Edge Information (REI)): *we consider the data transmission between two adjacent vehicles as a routing edge. On each routing edge of the data transmission, each packet is sent from the starting vehicle of this edge u, and arrives at the end vehicle of this edge v. These are just two traditional properties. Once the consideration of temporal information is included, two more temporal properties need to be included in the discussion, starting time t and duration d. These four properties of each routing edge are all routing edge information.*

At last, we introduce an efficient optimal routing algorithm in the temporal graph, and utilize the predicted routing as an input to achieve the optimal routing for each routing request. To demonstrate the superiority of our algorithm in terms of time cost, we also analyse the time complexity of each procedure of our routing algorithm. After that, we compare our proposed algorithm with other algorithms in terms of total time complexity and state them in the supplementary material.

A. Reasons for applying HMM

There are two key advantages of utilizing HMM in predicting routing as follows. First, it can foresee the future data transmission better. The reasonable HMM setting makes the model suitable to present a particular pattern of the vehicular network. The reason for this suitability is that all vehicles always move on the segments, where the movements of vehicles follow a certain pattern. Meanwhile, the intersections are the connecting points and the backbones of all segments. All segments are connected by intersections. Based on the statuses of all intersections, the entire traffic network can be easily

presented. This makes intersections particularly important in routing decisions [32]. With intersections as states and vehicles around the intersections as observations, the certain pattern can be learned in HMM model through training. Second, the prediction of HMM can improve the efficiency of our routing algorithm significantly since the temporal graph optimal path algorithm depends directly on the number of REIs. Here, HMM only explores the REIs in a certain sequence consisting of limited intersections instead of exploring them among all vehicles in the entire vehicular network. This can reduce the number of predicted REIs and guarantee the routing quality at the same time.

Indeed, compared with other popular prediction methods such as machine learning, HMM does not represent the state-of-the-art. However, this model has a unique advantage in the prediction of vehicular networks. Learning-based prediction methods such as machine learning cannot fulfill the complex demands of the vehicular network. We conclude the reasons as follows. First, since the routing prediction is different from the vehicle trajectory prediction, it highly depends on the current highly dynamic topology state and specific request demands [33]. Under this condition, there are quite a number of factors in the vehicular network affecting the prediction of routing. It is difficult to extract enough accurate features. For example, even if two static vehicles forward packets to each other, as time goes by, the statuses of all vehicles related to this data transmission will become completely different. In this case, the statuses of this starting node, the destination nodes, and all related vehicles need to be input as features. These large amount of features will inevitably cause overfitting. Second, the topology of vehicles is continually evolving. Offline learning is incapable of this task in this complex scenario without reasonable feature extraction. Meanwhile, the online learning-based prediction model cannot adjust itself in time to adapt to different networks because of high time cost and insufficient training data. In contrast, the prediction in HMM can adapt to the highly dynamic topology of the vehicular network efficiently. We only need to find the appropriate state and observation setting. After the training, the routing prediction pattern can be naturally presented in the probability distribution. Then the simple linear computations will be performed on the model with simple input. As a result, an accurate HMM can be achieved, which reflects the characteristics of the current vehicular network. Once the part of the network features changes, this model can also quickly adjust the relevant transition probabilities. We can conclude that the HMM is best option to be utilized as our prediction model.

B. Construction of HMM

In our HMM, we set intersections as hidden states and vehicles around the intersection as observations. When the controller receives a beacon message from a specific vehicle, the controller then judges which intersection this vehicle belongs to. Here, if one vehicle belongs to an intersection, it means that this vehicle is within a specific range of the intersection. We define the range of one intersection as follows.

DEFINITION 2 (Intersection Range): *We divide each segment originated from the intersection i into two parts averagely. The vehicles on the half segment, which is closer to the intersection i are defined as being within the intersection range of i.*

We consider all vehicles belong to an intersection as the vehicle subordinates of this intersection. By counting the statuses of vehicle subordinates at each intersection and the distance between every two intersections, the state transition probability distribution of the HMM can be achieved. There are three properties of an intersection we need to construct the state transition probability distribution.

(1) The position information of the intersection. This information can be easily fetched from the GPS of vehicles. To reduce the computational complexity of the following procedures, we can pre-process the position information of all intersections, and compute the distance between each intersection pair denoted as dis_{ij}. We then store it with an adjacency matrix.

(2) The subordinate vehicle-intersection information. This refers to the number of vehicles within the intersection range. In this context, each intersection range is divided into several parts regarding the segments originated from this intersection, where the number of parts is equal to the number of segments. If we would like to compute the transition probability from intersection i to intersection j, we should only consider the segment s_{ij} rather than other segments from i. There is a small probability that vehicles on other segments will be utilized to communicate with j. Each intersection range part only needs to take its responsibility to record the number and statuses of vehicles. We mainly use the vehicle density of the road segment to measure the stability of the link. Eq. (20) is utilized to compute the vehicle density.

$$den(s_{ij}) = \frac{V(s_{ij})}{l(s_{ij})} \tag{20}$$

where $den(s_{ij})$ denotes the vehicle density of the segment s_{ij} originated from intersection i. $V(s_{ij})$ represents the number of vehicles on the segment s_{ij} originated from the intersection i. Here, we need to notice that $V(s_{ij})$ is different from $V(s_{ji})$. More, each segment is divided into two parts averagely, where each part and vehicles on it belong to the intersection closer to them. Therefore, $V(s_{ij}) + V(s_{ji})$ is the real number of vehicles on the segment from the intersection i to j. Further, $l(s_{ij})$ means the length of the segment s_{ij}. All notations here will be presented in Table II.

(3) The historical intersection routing information. This means packet transmission between vehicles at two different intersections. We use the historical intersection routing index $rdex$ to represent this information. This index of each intersection pair is computed to construct the state transition probability distribution matrix. The index is achieved as formulated by Eq. (21).

$$rdex_{ij} = \sum_{h=0}^{p-1} \frac{r_{ij}^{t_h}}{(t_p - t_h)} \tag{21}$$

In Eq. (21), $redx_{ij}$ denotes the historical intersection routing index from intersection i to intersection j. t_h represents the

time value at time h. $h = 0$ means the time that the first historical REI happened. Here, t_p is the present time value. $r_{ij}^{t_h}$ is the valid routing value from intersection i to intersection j at time h. If a vehicle at intersection i was transmitting packets to the vehicle at intersection j at time h (i.e., the routing from i to j was valid), the value of $r_{ij}^{t_h}$ is 1; otherwise, it is 0. $redx_{ij}$ will rise with the number of data transmission between them in the past increasing. More, the farther a specific historic routing occurs from now, the less important it is.

With these three fundamental properties of the intersection, the state transition probability distribution of the HMM representing the current network can be constructed with dynamic programming and greedy strategies. First, we define the possibility of transiting from intersection i to j as a transition index from i to j, denoted by $tedx_{ij}$. The transition index is computed as Eq. (22).

$$tdex_{ij} = \begin{cases} \alpha * (k * (\frac{den(s_{ij})}{den_{max}(i)}) + \\ (1-k) * (\frac{rdex_{ij}}{rdex_{max}(i)})) + \\ (1-\alpha) * \frac{(dis_{i,des} - dis_{j,des})}{max(|dis_{i,des} - dis_{j,des}|)} & den(s_{ij}) > 0 \\ \\ 0 & den(s_{ij}) = 0 \end{cases} \tag{22}$$

If the vehicle density is 0, it means there is no vehicle on the segment s_{ij}. In this case, if we intend to transmit a packet between these two intersections, this transmission has a high probability of failure. Thus, the $tdex_{ij}$ is set to be 0 to avoid this link to transmit a packet.

Moreover, if the vehicle density is above 0, there are vehicles available for packet transmission. Hence, we define the transition probability in two parts, dynamic programming, and greedy. For the dynamic programming part, we take vehicle density $den(s_{ij})$ and the historical intersection routing index $rdex_{ij}$ as parameters. Higher vehicle density denotes more vehicles available for relaying data messages. In this case, the delivery ratio naturally rises. In another part, $rdex_{ij}$ indicates that there was a successful data transmission in the past. This link has a more probability of being stable and reliable now. $den_{max}(i)$ and $rdex_{max}(i)$ are the maximum vehicle density and the maximum historical intersection routing index among all segments starting from intersection i, respectively. k is the weight coefficient of the vehicle density. The bigger the k value is, the more critical the vehicle density information is. On the contrary, the importance of the historical intersection routing index grows with the decrease of the value of k. In order to make the model better iteratively update itself over time to adapt to the present vehicle network, we would like to set k to be larger. Since the network is constantly evolving, compared with the historical intersection routing information, the vehicle density information could be more reliable.

The second part is greedy, where $dis_{j,des}$ denotes the distance between intersection j and intersection des. Intersection des is the intersection to which the destination vehicle belongs. And $dis_{i,des}$ can also be understood in the same way. $dis_{i,des} - dis_{j,des}$ represents the degree of optimization for solving the routing computation problem after selecting this link. In simplicity, this shows how much closer away from the destination after taking j as the next-hop intersection. By taking this parameter into account, the transition index becomes more significant as the distance with the destination intersection decreasing. And, $max(|dis_{i,des} - dis_{j,des}|)$ is the maximum absolute value of the distance that can be approached to the destination from intersection i. Furthermore, α is the weight coefficient of the dynamic programming part. Here, we tend to set the value of α, which is the dynamic programming part, slightly larger than $1 - \alpha$. There are two reasons why we can adopt such a strategy. First is that we aim to put a higher priority on successful transmission. In other words, we intend to pursue the optimal routing path under the premise of guaranteeing link connectivity. Second is that we would like to avoid the local optimum by making the weight of the greedy part smaller.

Here, we need to make some explanations on the processing of prior knowledge. All the information we need for constructing the HMM is the position information of all vehicles ,intersections and the historical intersection routing information. The position information of vehicles and the historical intersection routing information is achieved from periodic data upload from the data plane. This process is independent of routing computation. Another part of the prior information, the position information of intersections has been stored in the control plane in advance before the routing process begins. In summary, the acquisition and processing of the prior information are simple in which it has little impact on the time cost of the main body of the routing algorithm.

The time complexity of this part is $O(N^2)$, where N is the number of intersections. Since the number of intersections in a given area does not change, the time cost of this part is constant in the routing computation. It does not change as the size of the network increases.

C. Prediction in HMM

After the computation of the $tedx_{ij}$ of each intersection pair, the state transition probability distribution of the HMM representing the current network has been constructed. With this model, the REI in the future is available. However, we need to predict the appropriate number of qualified REIs to guarantee the quality of the routing path and the efficiency of our algorithm at the same time. The number of predicted REIs should be limited. Meanwhile, all possibilities need to be included too. Thus, in this part, we will introduce the prediction method of our algorithm. First, we consider the state transition probability distribution as a graph where the vertex of this graph is the intersection. Then, we predict two intersection sequences, optimal path, and suboptimal path. The suboptimal path links to the optimal one at the appropriate nodes. Finally, we convert the states to the observations, i.e., intersection to vehicles. By linking the related vehicles together, an adequate number of qualified REIs can be achieved.

1) Hidden Sequence Generation: With the constructed state transition probability distribution, the next procedure is to find the intersection path with the biggest probability for transmitting packets. The goal of this procedure is similar to finding

the optimal path in graph theory. Thus, we utilize the classic optimal path algorithm, Dijkstra's algorithm, to accomplish this procedure. Several modifications will be applied to the model and the algorithm to adapt to this scenario.

The state transition probability distribution of the constructed HMM is a matrix, and can be considered as an adjacency matrix. The transition index can be regarded as the distance between two intersections. However, unlike the distance in reality, the larger the transition index is, the easier it is for this link to transmitting data. Thus, in order to better utilize the relevant knowledge in graph theory, we make the transition index as its reciprocal form. In this method, we make the transition index closer to the distance in reality, while the ratio remains the same. The corresponding adjacency matrix is constructed as these procedures. We first take the state transition probability distribution generated from the last procedure as input. Then all transition indexes between intersections will be traversed. If the index is above the average value of all transition indexes sourcing from the same intersection, we take it into account and consider it as an edge in the graph. The weight of this edge is the reciprocal form of this transition index. Otherwise, there is no edge corresponding to this index in the graph.

After that, we achieve the corresponding graph. Furthermore, we can formally enter the prediction stage. The first step is to generate the state sequence (i.e., intersection sequence). In this context, we choose the Dijkstra's algorithm to compute the optimal path from the source intersection to the destination intersection.

Because the size of the intersection matrix is much smaller than the vehicular network, the time cost of computation is small too. To explore all possibilities, we compute two paths, the optimal path and the suboptimal one. Then we link two paths together to achieve a mixed intersection path. In this way, we almost take all the possibilities into account and guarantee the appropriate number of qualified REIs at the same time. The algorithm detail is presented in Alg.1.

At the beginning, we take the adjacency matrix from the last step as the input. Then, we initialize three lists to store the result, including $optimal\_path$, $suboptimal\_path$, and $subgraph$ (Line 1). The $subgraph$ is applied to store the result of this algorithm, which is a mixed path with two optimal paths. Then we compute the optimal routing path on the graph G representing the state transition probability distribution with the Dijkstra's algorithm and store it in the $optimal\_path$ (Line 2). It is worth to mention that we use multiplication instead of plus to compute the cumulative distance in Dijkstra to fit the goal of the largest probability better. It is obvious that the computation of the total probability is based on the cumulative multiplication of various probabilities.

Since the number of intersections is limited, this procedure is computationally efficient. After that, we delete all intersections related to the $optimal\_path$ on the G except the source intersection and the destination intersection (Line 3 to 7). Then, we rerun the Dijkstra to get the suboptimal path and store it in $suboptimal\_path$ (Line 8). Now, two paths have been achieved. Then, we traverse $suboptimal\_path$. For each intersection, once an intersection in $optimal\_path$ is found as a closer node to the destination intersection, we link the intersection in the $suboptimal\_path$ to the closest one with it in the $optimal\_path$. We then store this link in the $subgraph$ and process the next intersection in the $suboptimal\_path$ (Line 9 to 16). At last, we insert the $optimal\_path$ and $suboptimal\_path$ into the $subgraph$. The mixed intersection path can be achieved.

Algorithm 1: Intersection sequence generation (G, s, d)

Input: A graph G representing the HMM, the intersection to which the source vehicle belongs s, the destination intersection d;
Output: a subgraph that is mixed with $optimal\_path$, $suboptimal\_path$, and their link;

1 **Initialization**: $optimal\_path \leftarrow []$, $suboptimal\_path \leftarrow []$, $subgraph \leftarrow []$;
2 $optimal\_path \leftarrow Dijkstra(G, s, d)$;
3 **foreach** *node* i *in* $optimale\_path$ **do**
4 **if** $i \neq s$ *or* $i \neq d$ **then**
5 delete i in G;
6 **end**
7 **end**
8 $suboptimal\_path \leftarrow Dijkstra(G, s, d)$;
9 **foreach** *node* i *in* $suboptimal\_path$ **do**
10 **foreach** *node* j *in* $optimal\_path$ **do**
11 **if** $d_{jd} < d_{id}$ **then**
12 insert edge e_{ij} into $subgraph$;
13 **continue**;
14 **end**
15 **end**
16 **end**
17 insert all edges in $optimal\_path$ into $subgraph$;
18 insert all edges in $suboptimal\_path$ into $subgraph$;
19 return $subgraph$;

The time complexities of adjacency matrix construction and Intersection sequence generation are $O(N \log N)$ and $O(N^2)$, respectively. Both algorithms depend on constant N, which is the number of intersections. Therefore, the enlargement of the vehicular network status will not affect the time cost of the routing computation.

2) Hidden States to Observation: In this part, we will introduce the final part of the prediction, conversion from the intersection to the vehicle. In this scenario, we need to predict the appropriate number of qualified REIs with the mixed intersection path we have achieved from the last procedure. At first, we explore the subgraph with the DFS method. Since some vehicles are not qualified enough to undertake the task of forwarding packets, we rank all the subordinate vehicles of it with Eq. (23) to quantify the vehicle qualification for each intersection we explore

$$q_x = \varphi \frac{\sum_{h=0}^{p-1} \frac{r_{vx}^{t_h}}{(t_p - t_h)}}{rdex_{max}} + (1-\varphi)\frac{1}{\sqrt{2\pi}}(\exp -|\frac{(dis_{v_x,ij} - \frac{l(s_{ij})}{2})^2}{2}|) \tag{23}$$

where q_x is the qualification index of the vehicle v_x. This can be considered as the observation symbol probability distribution. We consider this equation in two parts, the historical routing, and the distance. The historic routing part is almost the same as the one in the construction of HMM. The only difference is that we count the routing starting from the vehicle v_x. In the distance part, being too close or too far from the center of the intersection is not what we expect. If the vehicle gets too close, it is hard to transmit packets to the next intersection. If it is too far, it can hardly receive packets from the last intersection. Thus, vehicles in the middle part of the segment should have a higher qualification index. With the decrease of the distance between the vehicle and two ends of the segment, the qualification index gets smaller.For the same reasons as k in the construction of state transition probability matrix. We take the successful transmission as the highest priority, so we set $1-\varphi$ to be slightly larger. If the size of a segment is too large or the segment conditions are too complex (such as loop street), we need to pre-process this part of segments by dividing them into multiple parts. These segments can be girded through setting virtual intersections at the appropriate locations on the segments, such as the center of a long segment or a large curve. Since such complicated segment conditions only occasionally appear in the city, this will not cause an increment in the magnitude of the current number of intersections. If the size of segment is too small (i.e., smaller than the half of communication range), the connected intersection pair will be deleted, and one virtual intersection will be set in the middle of the segment. These virtual intersections play the same roles as the actual intersections playing in the HMM model stored in the controller. This method ensures that the actual road network can be restored as much as possible by the virtual graph stored in the control plane. Meanwhile, it is also ensured that vehicles on complex segments can correctly choose the relay nodes at the next intersection, regardless of whether the intersection is real or virtual.

Once q of this vehicle is above the two-thirds of the average value among all vehicles belong to this intersection, this vehicle is qualified and selected as a candidate for routing. Then, we make a full link between respective qualified vehicles at the pair of linked intersections in the $subgraph$. While forwarding REI prediction, we need to record the temporal information of the REI, following the end time (i.e., starting time + duration) of its predecessor. When the exploration reaches the destination vehicle, the quality and quantity of predicted REIs can be guaranteed in the end. Combined with these predicted REIs, a temporal graph representing the future routing of the current vehicular network can be achieved easily. The entire HMM prediction process is as shown in Fig. 2.

The time complexity of this part changes proportional to the increment in the size of the vehicular network. This makes each intersection has $\frac{2n}{3N}$ qualified subordinate vehicles. Thus, the time cost is $O((\frac{2n}{3N})^2)$ in linking all vehicles between two adjacent intersections and updating their temporal information. The total time of the hidden states to observation stage is $O((\frac{2n}{3N})^2 m)$. Here, m means the size of the predicted mixed intersection path, and it entirely depends on the size of the network and the distance between the source vehicle and the destination. Since the Dijkstra algorithm is utilized in the prediction part, m must be much smaller than N.

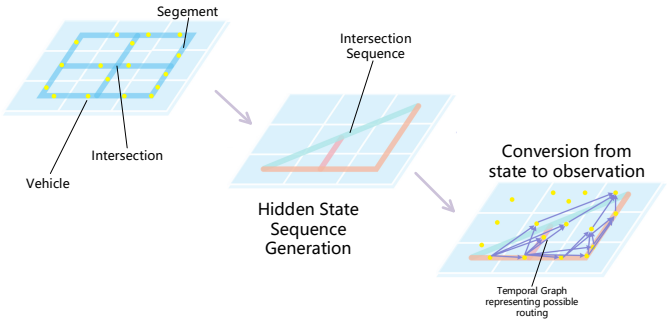


Fig. 2. The prediction process in HMM

D. Temporal Graph Algorithm

After the prediction, REIs are stored in the list ordered by their starting time. We take the optimal routing algorithm based on temporal information to find the optimal routing path we need. The predicted REIs are utilized as the input of this the optimal routing algorithm. Wu et al. propose four shortest path algorithms [9], including the earliest-arrival path, the latest-departure path, the shortest path, and the fastest path separately. Here, in this paper, we choose the first one, the earliest-arrival path, as our fundamental algorithm. It has the same goal with routing, which is transmitting the packet to the destination as early as possible. The temporal graph-based optimal path algorithm is an order-of-magnitude better than the existing algorithms in terms of efficiency [9], [14]. However, we need the information of each node on the optimal path as well. Thus, we set a predecessor node list for each node to record the optimal routing path.

The main idea of this algorithm is described as follows. First, we initialize a list to record the earliest arrival time of each vehicle. The earliest arrival time of the source vehicle is set to be 0. Others are ∞. We traverse all predicted REIs (u, v, t, d) with the sequence of starting time. If the starting time of one REI is later than the present earliest arrival time of u, that means this REI is qualified and reachable. Thus, this REI can enter the following judgment. If the end time $(t + d)$ is earlier than the current earliest arrival time of v, this means the REI is a better solution. All we need to do is to record this solution and update the earliest arrival time and the predecessor node list of v. After scanning all REIs in the list, the earliest arrival time from source to any other node can be easily achieved. Meanwhile, the node sequence of the optimal routing path can be achieved easily through scanning the predecessor node list recursively. The detailed pseudo code can be found in supplementary material. The time cost of this part is entirely dependent on the size of REIs as the input. It is $O((\frac{2n}{3N})^2 m)$, which is equal to the time complexity of the last procedure.

V. SIMULATION

In this section, we present details of the simulation, including parameters and evaluations. To evaluate our routing algorithm, we develop an open-source simulator in Python to meet the experimental demand of SDVNs (https://github.com/a824899245/SDVN-platform) [34]. Our simulation is based on a real map of Tiexi District, Shenyang City, China. The sizes of the map are set to be two levels, a small map $2686m \times 1494m$ and a large map $5193m \times 5863m$ (Specific satellite map images can be achieved in the Github project). Further, the covered locations of these two maps are the same. The map information is achieved from the OpenStreetMap [35]. Then, we apply SUMO [36] to generate the trajectory data of vehicle fitting on these maps. At last, we use our simulator to perform the data packet transmission in the vehicular networks. The transmission range is set as 500m. Even if the distance between the two vehicles is within range, high load and channel interference will still cause delay increase and even packet loss. These mechanisms have been integrated in the platform. Twenty routing requests are generated from random vehicles once per second. Also, the destination of the request is random. And we set average distance between the random source and the destination vehicle around half of the diagonal length of the map. Table III shows the parameter settings of our vehicular network and simulator. Each round of the simulation lasts for 300 seconds. Each round of simulation is repeated five times with different random seeds, while four routing algorithms use the same seeds.

We compare PT-GROUT to Dijkstra [37], HRLB [12], and PRHMM [20]. The first simulation counterpart is based on Dijkstra, which is the most popular algorithm of optimal routing in SDVNs. Authors tend to apply Dijkstra as their algorithms [11], [38], [39] directly or with modifications. Even though it is an efficient and accurate shortest path algorithm, it can miss essential temporal information of the vehicular networks. Besides Dijkstra, we also choose an SDVN routing scheme HRLB [12], as the other counterpart. HRLB is a hierarchical routing scheme with consideration of load balancing, which adopts a three-level architecture to compute the routing. HRLB also adopts the SDVN architecture and proposes a hierarchical routing scheme which is like the HMM in concept. Hence, we select this as a part of our counterparts. The third counterpart of PT-GROUT is PRHMM [20]. Similar to PT-GROUT, PRHMM also integrates prediction into the routing algorithm. In PRHMM, the vehicle predicts its position and the relationship with the destination vehicle by the HMM. While forwarding the data messages, each vehicle selects the next-hop vehicle according to the predicted delivery probability and delivery delay.

To compare the above routing schemes with considering various requirements for QoS, we choose four evaluation metrics, which are computation time cost, packet delivery ratio, delivery delay, and delivery delay jitter.

Computation time cost: The average run time for each routing request, this metric depends mainly on the time complexity of the routing algorithm.

TABLE III
SIMULATION PARAMETERS

Simulation Parameter Name	Value
Size of the simulation area	$2686m \times 1494m$ $5193m \times 5863m$
Intersections	$68/267$
Road segments	$116/457$
Number of vehicles	$100/200/300/400/500$
Vehicle velocity	$0 - 60km/h$
Vehicle transmission range	$500m$
Simulation duration	$300s$
Packet generation rate	20 packets per second
Data Packet Size	1024KB
Standards	IEEE 802.11p
α in Construction of HMM	0.6
k in Construction of HMM	0.6
φ in Hidden States to Observation	0.4

Packet delivery ratio: The ratio of delivered packets to the generated packets.

Delivery delay: The average end-to-end delay from the time when a packet is generated to the time when it is successfully delivered to its destination. This is the primary metric to show the quality of the routing policy.

Delivery delay jitter: The variance of the delay of each successful packet transmission. This presents the volatility of the quality of the routing path. In some respects, it presents the quality of the routing policy.

The average value of four metrics will be presented in the simulation figures. According to the five-rounded different simulations, the 90% confidence interval for each metrics will also be presented in the figures around the average value. The detailed mathematical definitions of these metrics are in Section II(B).

A. Routing Performance

This part provides the evaluation of the performance of our algorithm on the city map with a small geographical area $(2686m \times 1494m)$ and a large area $(5193m \times 5863m)$. As shown in Fig. 3(a) and Fig. 4(a), we compare the performance of these four routing algorithms in terms of computation time cost. It is worth noting that the computation time cost denotes the time complexity of computing the routing strategy in the controller. Hence, we do not consider the delivery delay. In almost all cases, our algorithm outperforms others all the time. PT-GROUT is efficient because of the limited number of intersections, in which the information of intersections has been pre-processed at the status updating stage. Also, the temporal-graph optimal routing algorithm runs in a linear time. The efficiency advantage of PT-GROUT grows with

the increasing number of nodes. For the low-density scenario (i.e., less than 200 nodes in the large map), the computation efficiency of Dijkstra could be similar to that of PT-GROUT since the number of intersections is close to the number of vehicles. And we also need to utilize Dijkstra to achieve the intersection sequence. Thus, in such scenario there are no significant differences in efficiency between PT-GROUT and Dijkstra. However, with the increment in the number of vehicles, PT-GROUT can improve the computation efficiency of the controller significantly. Since the time cost of this entire HMM-related part of the routing algorithm mainly relies on the number of intersections, it will remain stable with the increment of the vehicle density. This is a remarkable improvement in terms of efficiency. All SDVN routing algorithms remain stable in terms of the computation cost as shown in the confidence intervals. Among all, PT-GROUT achieves the most stable performance since the size of intersections remains the same. The instability of PRHMM is attributed to the distribution method. The lack of global vision and the uneven distribution of computing resources among vehicles could be the main reason.

Due to the restriction on the simulation, the maximum number of vehicles is set as 500. In a real-life scenario, the number of vehicles in peak hours will be much higher than this number. The average length of the segment is 300m. Moreover, the average vehicle density is 50 vehicles per km. We find 68 intersections on the small map, in which the number of segments can reach 110. After the computation, the number of vehicles can reach 1650. This will give substantial computation burdens to these routing algorithms whose computation efficiency highly relies on the number of vehicles. However, the larger number of vehicles will not have much impact on the efficiency of our algorithm. The time cost of the HMM-related part is only related to the number of intersections, which is much smaller than the number of vehicles in real life. Meanwhile, the Hidden states to Observation part and the following Temporal Graph part is only partly related to vehicle density and the time complexity of them will be satisfying. Thus, PT-GROUT can reduce the computation burden of the controller greatly. The corresponding time complexity analysis of all routing algorithms is presented in the supplementary material.

The reasons behind that the computation time cost on the large map is higher than that on the small map can be explained as follow. Even though the number of vehicles is the same, the corresponding graph will become more sparse, as the map area increases. Thus, since the communication distance is fixed, the routing algorithm only needs to process a smaller number of edges to complete the routing discovery.

In Fig. 3(b) and Fig. 4(b), we present a comparison of the delivery ratio. Our algorithm demonstrates a high delivery ratio at five levels of different numbers of vehicles. PT-GROUT outperforms the other three algorithms in terms of the delivery ratio. There are two main reasons for the better performance of PT-GROUT, the adequate number of possible predicted REIs and the consideration on the vehicle density. Through our HMM construction and prediction methods, PT-GROUT has an appropriate number of qualified predicted REIs for the final discovered routing path. Among these REIs, all possibilities of routing can be included, and the size is limited according to the size of the map to improve the routing path computation efficiency. Meanwhile, the consideration of vehicle density ensures that these predicted REIs are available in the future with great probability. These two measurements mostly ensure the reachability of the routing process.

An unusual phenomenon is the low packet delivery ratios of all four algorithms at the low-density scenario in the large area scenario. The reason is that one hundred vehicles are too sparse for the area of $5193m \times 5863m$. In this case, it is hard to find the next-hop vehicle to forward data messages in such a sparse scenario, which leads to packet loss. Thus, in an overly sparse network, all algorithms do not perform well. The low packet delivery ratio increases gradually with the increment in the number of vehicles, and achieve good and stable performance in the small map. Meanwhile, the network also becomes more complicated with the increase of vehicle density. Some routing algorithms, such as HRLB and PRHMM, cannot analyze the vehicular network thoroughly to find a reasonable routing path stably, which ultimately leads to the failure of routing process and the decreasing of delivery ratio. Under all these conditions, the packet delivery ratio of PT-GROUT remains the highest quality in any scenario.

Fig. 3(c), Fig. 3(d), Fig. 4(c) and Fig. 4(d) show delay, and jitter, respectively. PT-GROUT demonstrates better performance than the other three routing algorithms in terms of delay. The average delivery delay becomes longer with the increment of vehicles under other two SDVN routing algorithms. The reason could be the increment of the relay vehicles. Other SDVN routing algorithms tend to compute the routing path with more vehicles when the vehicle topology becomes dense. The total forwarding time climbs up with the increment of relay vehicle. PT-GROUT will only select a limited number of vehicles as relays under any circumstances, because the number is determined by the length of the hidden state sequence, and unrelated to the number of vehicles. The HMM computation contributes to this improvement in term of delay performance. Thus, the delay of routing path computed by PT-GROUT remain small and stable. Meanwhile, the increment of vehicles does not seem to have a significant impact on the routing computation strategy of PRHMM. Thus, PRHMM performs better than other two SDVN algorithms in terms of delay in the dense network. However, due to the lack of global view in the distributed routing algorithms, the global optimal routing cannot be computed as sometimes the local optimum will be triggered by PRHMM with the increase of the number of vehicles. The above will inevitably lead to the increment and fluctuation of routing delay.

Besides, for delay jitter, Dijkstra and PT-GROUT show good performance compared with others since they always compute the optimal routing path among all four algorithms. For PT-GROUT, the stable number of relay vehicles restricted by the intersection sequence contributes to this result. The disadvantages of distributed nature of these routing algorithms are obvious in terms of delay jitter. The limited vision of the vehicles makes PRHMM unable to compute the qualified routing stably from time to time.

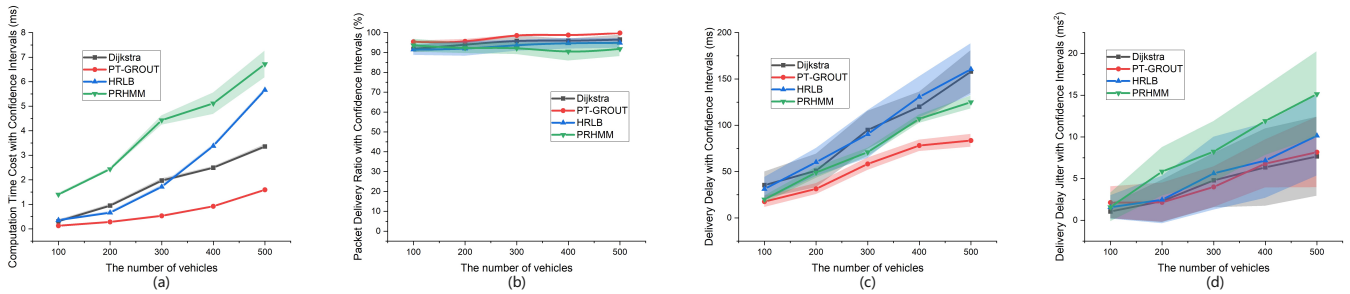


Fig. 3. Four metrics versus the number of vehicles on the small small-area map. (a) Computation time cost with confidence intervals (ms), (b) Packet delivery ratio with confidence intervals $(\%)$, (c) Packet delivery delay with confidence intervals (ms), (d) Delivery delay jitter with confidence intervals (ms^2).

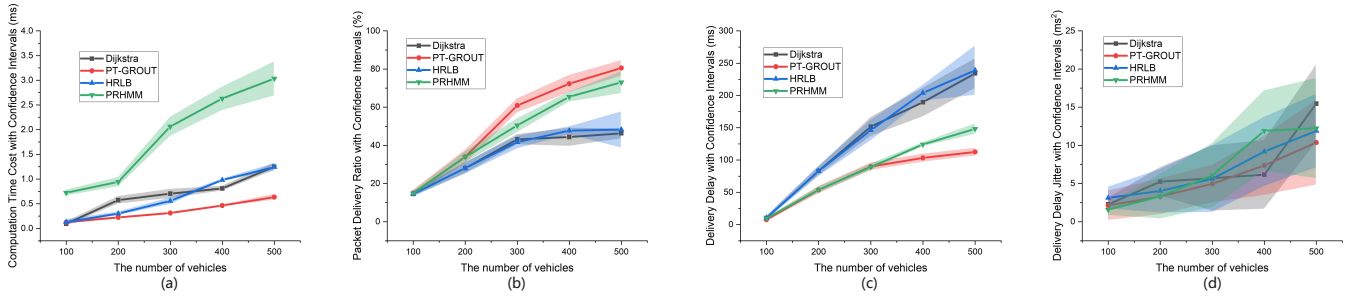


Fig. 4. Four metrics versus the number of vehicles on the large-area map. (a) Computation time cost with confidence intervals (ms), (b) Packet delivery ratio with confidence intervals $(\%)$, (c) Packet delivery delay with confidence intervals (ms), (d) Delivery delay jitter with confidence intervals (ms^2).

By combining the analysis above, we can draw that our proposed algorithm can compute the routing path of good quality in the scenarios with a various number of nodes efficiently. The most apparent reason for the better quality of the routing path is the higher delivery ratio of PT-GROUT. Moreover, through the HMM prediction, all the possible routes have been taken into consideration. The quality and quantity of them are guaranteed. In quality, all routing possibilities can be found among predicted REIs. In quantity, the number of REIs is limited by the intersection to ensure the high computation efficiency of PT-GROUT. Since the number of intersections is only related to the size of the map, the computation cost and the delay of PT-GROUT are good and stable. At last, the efficient temporal graph routing algorithm also contributes to this result. This algorithm can compute the absolute optimal routing path among these predicted routing efficiently. The results prove the high quality and computation efficiency of the computed routing.

At last, we would like to present the frequency of using GPSR as an alternative algorithm in PT-GROUT. The unit we use here is the GPSR occurrence, which is calculated as the number of times that GPSR is utilized as an alternative algorithm / the total number of routing computation processes. As we can see in Fig.5, the GPSR occurrence also varies with the vehicle density and decreases with the increase of the vehicle density overall. It is getting harder to achieve the feasible routing path as the vehicle density becomes lower. Thus, our work intends to provide more ways to find available routing paths, including using GPSR as an alternative algorithm. Meanwhile, for the same reason, GPSR occurrences on the large maps are generally greater than those on the small

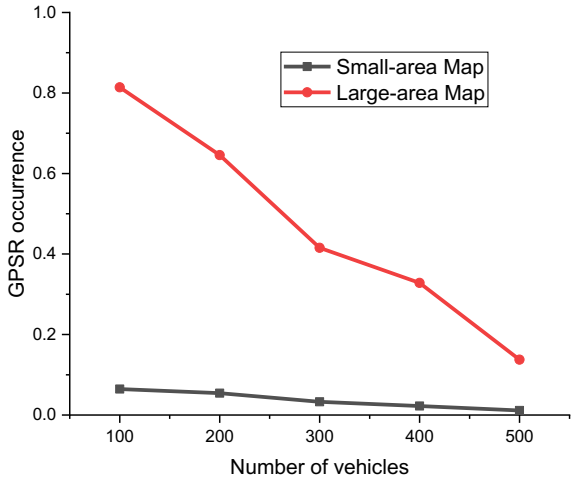


Fig. 5. GPSR occurrence versus the number of vehicles on the both small and large-area map.

map. This is also because the vehicle density on the large maps is much lower than that on the small maps.

B. Summary of Simulation

Referring to the two different sizes of real maps in Shenyang, China, we design a series of simulations to evaluate the performance of PT-GROUT on routing computation and prediction. In terms of all four metrics for evaluating the routing quality, PT-GROUT always remains at a high level in different scenarios. Under four metrics, PT-GROUT shows better and stabler performance than others in most cases. Under other scenarios, at least it is similar to the algorithm

with the best performance. The simulations show that PT-GROUT can compute the routing path with high quality most efficiently and stably. The routing quality can be guaranteed while the time complexity can be significantly reduced. After all the simulations, it can be proven that PT-GROUT is qualified and efficient enough to handle redundant routing requests in the increasingly complex vehicular network.

VI. Conclusion

In this paper, we propose a prediction-based vehicular routing algorithm in SDVNs to improve the routing performance and the computation efficiency at the same time. We integrate two essential and novel strategies into our algorithm, HMM for prediction, and temporal graph for computation. In this context, the hidden states of the HMM are intersections, where the observations are the vehicles around the intersection. Reasonable settings of HMM make the quality and quantity of predicted future REIs trustable. All possible routing selections are available, and the computation efficiency can be ensured by the limitation on the number of predicted REIs. Also, we construct the temporal graph with the predicted REIs, making it more adaptive to the real-world scenarios. The efficient temporal-graph optimal routing algorithm reduces the computation burden in the control plane significantly. We have performed extensive simulations to compare PT-GROUT with several other state-of-the-art schemes in computation efficiency, delivery ratio, delivery delay, and delay jitter. PT-GROUT shows better performance than other algorithms in most routing scenarios under different evaluation metrics.

Acknowledgment

This paper is partly supported by the Young and Middle-aged Science and Technology Innovation Talent Support Plan of Shenyang (RC190026), the Liaoning Natural Science Foundation (2020-MS-237), and the Liaoning Provincial Department of Education Science Foundation (JYT19052).

References

[1] J. Cheng, J. Cheng, M. Zhou, F. Liu, S. Gao, and C. Liu, "Routing in Internet of Vehicles: A Review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 5, pp. 2339–2352, 2015.

[2] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan 2015.

[3] B. A. A. Nunes, M. Mendonca, X. Nguyen, K. Obraczka, and T. Turletti, "A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks," *IEEE Communications Surveys Tutorials*, vol. 16, no. 3, pp. 1617–1634, Third 2014.

[4] K. S. Kalupahana Liyanage, M. Ma, and P. H. Joo Chong, "Controller placement optimization in hierarchical distributed software defined vehicular networks," *Computer Networks*, vol. 135, pp. 226–239, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128618300951

[5] K. Wang, H. Yin, W. Quan, and G. Min, "Enabling Collaborative Edge Computing for Software Defined Vehicular Networks," *IEEE Network*, vol. 32, no. 5, pp. 112–117, 2018.

[6] L. Zhao et al., "Routing Schemes in Software-Defined Vehicular Networks: Design, Open Issues and Challenges," *IEEE Intelligent Transportation Systems Magazine*, 2019.

[7] L. Zhao, W. Zhao, A. Al-Dubai, and G. Min, "A novel adaptive routing and switching scheme for software-defined vehicular networks," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–6.

[8] F. A. Silva, A. Boukerche, T. R. Silva, E. Cerqueira, L. B. Ruiz, and A. A. Loureiro, "Information-Driven Software-Defined Vehicular Networks: Adapting Flexible Architecture to Various Scenarios," *IEEE Vehicular Technology Magazine*, vol. 14, no. 1, pp. 98–107, 2019.

[9] H. Wu, J. Cheng, S. Huang, Y. Ke, and Y. Xu, "Path Problems in Temporal Graphs," *Proceedings of the VLDB Endowment*, vol. 7, no. 9, pp. 721–732, 2014.

[10] V. Kostakos, "Temporal Graphs," *Physica A: Statistical Mechanics and its Applications*, vol. 388, no. 6, pp. 1007–1023, 2009.

[11] M. S. Rayeni and A. Hafid, "Routing in Heterogeneous Vehicular Networks using an Adapted Software Defined Networking Approach," in *2018 Fifth International Conference on Software Defined Systems (SDS)*, April 2018, pp. 25–31.

[12] Y. Gao, Z. Zhang, D. Zhao, Y. Zhang, and T. Luo, "A Hierarchical Routing Scheme With Load Balancing in Software Defined Vehicular Ad Hoc Networks," *IEEE Access*, vol. 6, pp. 73 774–73 785, 2018.

[13] L. Zhao, Z. Li, J. Li, A. Al-Dubai, G. Min, and A. Y. Zomaya, "A Temporal-Information-Based Adaptive Routing Algorithm for Software Defined Vehicular Networks," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–6.

[14] H. Wu, Y. Huang, J. Cheng, J. Li, and Y. Ke, "Reachability and Time-Based Path Queries in Temporal Graphs," in *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE, 2016, pp. 145–156.

[15] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, Feb 1989.

[16] G. Karagiannis, O. Altintas, E. Ekici, G. Heijenk, B. Jarupan, K. Lin, and T. Weil, "Vehicular Networking: A Survey and Tutorial on Requirements, Architectures, Challenges, Standards and Solutions," *IEEE Communications Surveys Tutorials*, vol. 13, no. 4, pp. 584–616, Fourth 2011.

[17] B. Karp and H.-T. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," in *Proceedings of the 6th annual international conference on Mobile computing and networking*. ACM, 2000, pp. 243–254.

[18] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in *Proceedings WMCSA'99. Second IEEE Workshop on Mobile Computing Systems and Applications*, Feb 1999, pp. 90–100.

[19] V. Namboodiri and L. Gao, "Prediction-Based Routing for Vehicular Ad Hoc Networks," *IEEE Transactions on Vehicular Technology*, vol. 56, no. 4, pp. 2332–2345, July 2007.

[20] L. Yao, J. Wang, X. Wang, A. Chen, and Y. Wang, "V2X Routing in a VANET Based on the Hidden Markov Model," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, pp. 889–899, March 2018.

[21] X. Yan, P. Dong, X. Du, T. Zheng, J. Sun, and M. Guizani, "Improving flow delivery with link available time prediction in software-defined high-speed vehicular networks," *Computer Networks*, vol. 145, pp. 165–174, 2018.

[22] L. Zhao, Z. Bi, M. Lin, A. Hawbani, J. Shi, and Y. Guan, "An intelligent fuzzy-based routing scheme for software-defined vehicular networks," *Computer Networks*, vol. 187, p. 107837, 2021.

[23] S. Scellato, I. Leontiadis, C. Mascolo, P. Basu, and M. Zafer, "Evaluating Temporal Robustness of Mobile Networks," *IEEE Transactions on Mobile Computing*, vol. 12, no. 1, pp. 105–117, Jan 2013.

[24] L. Qiao, Y. Shi, and S. Chen, "An Empirical Study on the Temporal Structural Characteristics of VANETs on a Taxi GPS Dataset," *IEEE Access*, vol. 5, pp. 722–731, 2017.

[25] L. Zhao, X. Li, B. Gu, Z. Zhou, S. Mumtaz, V. Frascolla, H. Gacanin, M. I. Ashraf, J. Rodriguez, M. Yang, and S. Al-Rubaye, "Vehicular Communications: Standardization and Open Issues," *IEEE Communications Standards Magazine*, vol. 2, no. 4, pp. 74–80, December 2018.

[26] T. S. Rappaport *et al.*, *Wireless communications: principles and practice*. prentice hall PTR New Jersey, 1996, vol. 2.

[27] J. Goldhirsh and W. J. Vogel, "Handbook of propagation effects for vehicular and personal mobile satellite systems," *NASA Reference Publication*, vol. 1274, pp. 40–67, 1998.

[28] M. Killat and H. Hartenstein, "An empirical model for probability of packet reception in vehicular ad hoc networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2009, pp. 1–12, 2009.

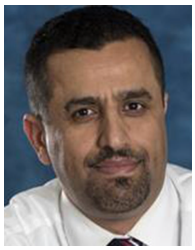
[29] C. Bovy, H. Mertodimedjo, G. Hooghiemstra, H. Uijterwaal, and P. Van Mieghem, "Analysis of End-to-End Delay Measurements in Internet," in *Proc. of the Passive and Active Measurement Workshop-PAM*, vol. 2002. sn, 2002.

[30] K. K. Sudheera, M. Ma, G. M. N. Ali, and P. H. J. Chong, "Delay Efficient Software Defined Networking Based Architecture for Vehicular

Networks," in *2016 IEEE International Conference on Communication Systems (ICCS)*. IEEE, 2016, pp. 1–6.
[31] A. Hawbani, X. Wang, A. Al-Dubai, L. Zhao, O. Busaileh, P. Liu, and M. A. Al-qaness, "A Novel Heuristic Data Routing for Urban Vehicular Ad-hoc Networks," *IEEE Internet of Things Journal*, 2021.
[32] M. A. Togou, A. Hafid, and L. Khoukhi, "SCRP: Stable CDS-based routing protocol for urban vehicular ad hoc networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 5, pp. 1298–1307, 2016.
[33] C. Wu, Z. Liu, D. Zhang, T. Yoshinaga, and Y. Ji, "Spatial Itelligence toward Trustworthy Vehicular IoT," *IEEE Communications Magazine*, vol. 56, no. 10, pp. 22–27, 2018.
[34] L. Zhao, G. Han, Z. Li, and L. Shu, "Intelligent Digital Twin-Based Software-Defined Vehicular Networks," *IEEE Network*, vol. 34, no. 5, pp. 178–184, 2020.
[35] M. Haklay and P. Weber, "OpenStreetMap: User-Generated Street Maps," *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12–18, 2008.
[36] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent Development and Applications of SUMO - Simulation of Urban MObility," *International Journal On Advances in Systems and Measurements*, vol. 5, no. 3&4, pp. 128–138, December 2012. [Online]. Available: http://elib.dlr.de/80483/
[37] R. Bhandari, *Survivable Networks: Algorithms for Diverse Routing*. Springer Science & Business Media, 1999.
[38] G. Secinti, P. B. Darian, B. Canberk, and K. R. Chowdhury, "SDNs in the sky: Robust End-to-End Connectivity for Aerial Vehicular Networks," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 16–21, Jan 2018.
[39] S. Yang, J. Liu, and X. Yan, "Flexible Routing-Proactive Updating Mechanism for Software Defined Vehicle Networks," in *2018 10th International Conference on Wireless Communications and Signal Processing (WCSP)*, Oct 2018, pp. 1–6.

Liang Zhao [M] is an associate professor at Shenyang Aerospace University, China. He received his Ph.D. degree from the School of Computing at Edinburgh Napier University in 2011. Before joining Shenyang Aerospace University, he worked as associate senior researcher in Hitachi (China) R&D from 2012 to 2014. His research interests include ITS, VANET, WMN and SDN. He has published more than 100 high-quality peer-reviewed papers. He served as the Chair and Co-Chair of more than 20 international conferences and workshops such as 2021 IEEE TrustCom (Program Co-Chair) and 2020 IEEE IUCC (General Co-Chair). He is/has been a guest editor of journals such as IEEE Transactions on Network Science and Engineering, Springer Journal of Computing. He was the recipient of the Best/Outstanding Paper Awards at IEEE IUCC 2015, ACM MOMM 2013, and IEEE ISPA 2020.

Zhuhui Li received the B.S. and M.S. degree in Computer Science and Technology from Shenyang Aerospace University, China. Currently he is a Ph.D. postgraduate student in the Department of Mathematics and Computer Science within the College of Engineering, Mathematics and Physical Sciences at the University of Exeter, United Kingdom. His research interests mainly include VANETs, FANETs, SDVN and Temporal Graph.



Ahmed Y. Al-Dubai [SM] is Professor of Networking and Communication Algorithms in the School of Computing at Edinburgh Napier University, UK. He received the PhD degree in Computing from the University of Glasgow in 2004. His research interests include Communication Algorithms, Mobile Communication, Internet of Things, and Future Internet. He received several international awards.

Geyong Min is a professor of high-performance computing and networking in the Department of Mathematics and Computer Science within the College of Engineering, Mathematics and Physical Sciences at the University of Exeter, United Kingdom. He received his Ph.D. degree in computing science from the University of Glasgow, United Kingdom, in 2003, and his B.Sc. degree in computer science from Huazhong University of Science and Technology, China, in 1995. His research interests include future Internet, computer networks, wireless communications, multimedia systems, information security, high-performance computing, ubiquitous computing, modeling, and performance engineering.

Jiajia Li [M] is an associate professor at Shenyang Aerospace University, China. She received the B.S. degree from Northeast Normal University in software in 2008 and received the M.S. and Ph.D. degrees in Computer Science from Northeastern University in 2010 and 2014. Her research interests include mobile data management, spatial-temporal database, intelligent transportation system.

Ammar Hawbani [M] received the B.S., M.S. and Ph.D. degrees in Computer Software and Theory from the University of Science and Technology of China (USTC), Hefei, China, in 2009, 2012 and 2016, respectively. Currently, he is a postdoctoral researcher in the School of Computer Science and Technology at USTC. His research interests mainly in WSN and WBAN.

Albert Y. Zomaya [F] is a Chair Professor and director of the Centre for Distributed and High Performance Computing at the University of Sydney. He has published more than 600 scientific papers and is an author, co-author, or editor of more than 20 books. He is the Editor-in-Chief of IEEE Transactions on Sustainable Computing and ACM Computing Surveys and serves as an Associate Editor for several leading journals. He is a Fellow of IEEE, AAAS, IET, and member of Academia European.