Greta: Towards A General Roadside Unit Deployment Framework

Xianjing Wu, Zhidan Liu, *Member, IEEE,* Zhenjiang Li, *Member, IEEE,*
Shengjie Zhao, *Senior Member, IEEE*

Abstract—As an essential component, roadside units (RSUs) play an indispensable role in realizing Vehicle-to-Everything (V2X) by seamlessly connecting various intelligent devices and vehicles. To facilitate the construction of V2X, much research has been done in designing effective RSU deployment strategies. However, most of these efforts are largely limited by design utility and deployment scalability. To address the limitations of previous works, this paper proposes a general RSU deployment framework, *Greta*, which can evaluate candidate deployment sites from different perspectives with rich input data, and satisfy different requirements on optimization metrics. To this end, we model the general RSU deployment problem as a customized reinforcement learning (RL) problem that intelligently explores the deployment environment to find a good deployment strategy. Specifically, we design an effective data profiling network to extract features from multi-modality input data. These extracted features are gradually weighted, fused, and encoded as part of the state representation of the RL model. We further design new reward functions considering various deployment metrics and propose an action space pruning scheme to speed up model training. We implement a prototype system of *Greta* and extensively evaluate its performance using real-world data. The results show *Greta* achieves remarkable performance gains compared to recent RSU deployment methods.

Index Terms—Vehicle-to-Everything, Roadside Unit Deployment, Reinforcement Learning.

1 Introduction

VEHICLE-to-Everything (V2X) is a promising communication technology that can enable a variety of emerging smart transportation applications (e.g., automatic driving [1], traffic optimization [2] and in-car entertainment [3]) and an important way to reduce traffic accidents and fleet operating costs in future transportation systems [4]. As communication gateways, *roadside units* (RSUs) play an indispensable role in realizing the V2X concept by seamlessly connecting various devices and vehicles [5]. Specifically, RSUs can collect information from sensing devices, traffic infrastructure, and surrounding intelligent connected vehicles, upload this information to the V2X platform through wired or wireless channels, and distribute traffic information to relevant vehicles [6]. With a wide and effective deployment of RSUs on the road network, the efficiency and coverage of information exchange in V2X can be greatly improved, leading to better traffic control, road safety, and informative roadway services [7].

- *X. Wu is with the Department of Computer Science, City University of Hong Kong, Hong Kong, China, and with the College of Electronic and Information Engineering, Tongji University, Shanghai, China, 201804. E-mail: xianjinwu2-c@my.cityu.edu.hk.*
- *Z. Liu is with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. E-mail: liuzhidan@szu.edu.cn.*
- *Z. Li is with the Department of Computer Science, City University of Hong Kong, Hong Kong, China. E-mail: zhenjiang.li@cityu.edu.hk.*
- *S. Zhao is with the School of Software Engineering, Tongji University, Shanghai, China, 201804, and with the Engineering Research Center of Key Software Technologies for Smart City Perception and Planning, Ministry of Education, and with the Key Laboratory of Embedded System and Service Computing, Ministry of Education. E-mail: shengjiezhao@tongji.edu.cn.*
- *Zhidan Liu and Shengjie Zhao are the corresponding authors.*

Given the huge potential of V2X, many countries, such as China, the United States, and Japan, have developed visionary plans to actively install RSUs to promote the construction of future V2X-enabled intelligent transportation systems [8], [9]. However, RSUs are often characterized by high deployment costs [10]. Therefore, given a limited budget (e.g., the total number of RSUs or deployment costs), how to effectively and efficiently deploy the RSUs to maximize their utility is a crucial and practical problem [11]. In the literature, many research efforts have been made for the deployment of RSUs, but they have two main limitations:

(*i*) *Design utility*. Previous studies have mainly focused on optimizing some specific deployment objectives, which require predefined optimization metrics (e.g., vehicle connectivity [12], road coverage [13], or communication quality [14]), tailored formulations and specialized solutions. They make meaningful pioneer contributions to advancing RSU deployments, but a major limitation is that their solution is highly constrained by the targeted metric and/or formulation. Such an end-to-end solution limits its utility and often requires new designs to decide on deployment strategies when the deployment scenarios are different.

(*ii*) *Deployment scalability*. On the other hand, deploying RSUs in practice is not a one-time process, and often requires the gradual addition of RSUs on the road network. During long-term deployments, optimization metrics may be adjusted. In addition, as more and more sensors are installed on RSUs [15], [16], [17], joint use of multiple metrics may also be required in the future. Holistic consideration of various optimization metrics, together with deployed RSUs to well plan future RSU positions, has been rarely studied in previous works, which however is an inevitable problem in practice.

Therefore, we propose a general RSU deployment framework, called *Greta*, in this paper to address the aforementioned limitations. Rather than relying on hand-crafted input features and specific optimization metrics, our framework incorporates an *input information library* consisting of various input data, e.g., geometric map data and mobility data, and an *output metric library*, including a set of widely used performance metrics, e.g., road/traffic coverage or communication quality indicators. The framework can automatically learn to decide which set of input data to use and how to apply them to fulfill the output requirements to guide RSU deployment (potentially involving multiple deployment metrics). A significant advantage of *Greta* is that both the input and output libraries are adjustable and extensible, making it possible to efficiently decide or update deployment requirements. In addition to deploying RSU from scratch, *Greta* also supports incremental deployments. Given a set of deployed RSUs, it can evaluate the utility for any new output metric requirements and guide the deployment of additional RSUs on top of the existing ones. To harvest these benefits, we address the following challenges.

First, the input data source candidates in the input information library usually have diverse modalities and different impacts on various optimization metrics. How to properly characterize and fuse multi-modality data to obtain effective input for RSU deployment needs to be carefully studied. To address this issue, we propose an effective data profiling network to extract features from each candidate input data source. The extracted features are then gradually weighted, fused, and encoded as a comprehensive representation of the deployment environment conditioned by the desired optimization metric(s) in a latent state space, which is further used by *Greta* to derive the deployment strategy.

Second, there are enormous locations in a road network where RSUs can be possibly deployed. The search space for suitable deployment sites is huge, which requires extensive computations to determine the optimal solution. On the other hand, when solving the deployment problem, we lack ground-truth labels to quantify the quality of this deployment. To address this problem, we employ deep reinforcement learning (DRL) to gradually explore the search space to find a good deployment strategy [18]. However, DRL is just a framework, and we thus customize it for the general RSU deployment in this paper. In particular, we leverage our extracted features to construct DRL states, design a series of reward functions for various deployment metrics, and propose an action space pruning scheme to avoid unnecessary explorations to speed up the model training.

We develop a prototype system of *Greta* and examine its performance based on a large GPS trajectory dataset in the downtown area of Chengdu City, China. As the initial implementation, we realize the input information library with the road map and one-month GPS trajectory data collected from thousands of contract vehicles of Didi Chuxing [19] and instantiate the output metric library with various deployment metrics, including road coverage, traffic coverage, and a combination of these two metrics. Extensive results show that, based on the road coverage metric, *Greta* achieves 18.5-40.0% performance gains compared to recent RSU deployment methods and up to 7.2% performance gains compared with the Simulated Annealing search method. In summary, this paper makes the following contributions:

- We propose a general RSU deployment framework named *Greta*. It can be extensible to various input data sources and deployment metrics, which are inevitable in future RSU deployments to realize different V2X services.
- We identify two key challenges in designing *Greta*, and propose an effective data profiling network to adaptively fuse multiple input data sources in the latent state space and customize a DRL model to intelligently explore the best deployment strategy.
- We develop a prototype system and evaluate it using real-world data. Extensive experiments demonstrate the effectiveness of our system. Compared to existing RSU deployment methods, our system can achieve promising performance gains on various deployment metrics.

The rest of the paper is organized as follows: Section 2 reviews the related works. Section 3 introduces the RSU deployment background and motivates our study on the general deployment framework. Section 4 presents the design details of *Greta*. We implement *Greta* and evaluate its performance in Section 5, following with the discussion of *Greta* in Section 6. Finally, Section 7 concludes this paper.

2 Related Work

As a core infrastructure of V2X, RSU deployment has attracted significant attention in recent years. Initially, V2X was used in pilot projects on highways, resulting in early RSU deployment works focusing on one-dimensional modeling [20], [21], [22], [23], [24], which fit the highway condition. However, as urbanization has developed, there is a need to deploy V2X services in complex urban areas. As a result, recent works have focused on practical two-dimensional modeling of RSU deployment problems [25], [26]. We classify these works from the following two perspectives, i.e., optimization objective and modeling, and discuss the most related works as follows.

Optimization objective: Previous works have targeted different deployment objectives or requirements, which can be divided into three categories. 1) *Coverage:* Coverage includes spatial and temporal coverage. For example, Zhang *et al.* consider the coverage area as one important objective for their RSU deployment optimization [25], while Mokhtari *et al.* consider the V2I connection duration of the vehicles within the RSU covered area as the objective [12]. Kim *et al.* consider both temporal and spatial coverage [27]. 2) *Service:* Many works aim to develop deployment strategies by optimizing the services provided by RSUs, including resource allocation such as communication, computing, and caching. Some of them use communication indicators as the optimization objective for RSU deployment [28], [29]. For example, Wu *et al.* find the optimal deployment scheme by maximizing the aggregate throughput in the network under RSU coverage [21], while Mehar *et al.* optimize deployment to reduce delay [30]. Although computing and caching modules are not yet standard on RSUs, some studies have considered them in the optimization of RSU deployment [31],

[32], [33]. 3) *Others:* A few works have different concerns on the RSU deployments. Specifically, Salari *et al.* investigate the optimal deployment of RSUs for path-level traffic flow reconstruction.Jiang *et al.* [34] and Sreya *et al.* [35] study the RSU deployment problem under the V2I-based traffic prediction application.

These research works, however, usually focus on optimizing some specific deployment objectives, and thus are highly constrained by the targeted metrics. The end-to-end designs limit their utility for these emerging V2X services that may have varying deployment requirements. Different from them, the flexible design of *Greta* allows RSU deployments to adapt to various requirements of V2X services.

Modeling: Previous works vary in problem modeling, along with different solutions relying on various optimization techniques, including linear programming [36], nonlinear programming [37], integer programming [21], binary programming [38], mixed integer programming [39], nonconvex optimization [40], dynamic programming [41], among others. Due to the large search space, RSU deployment problem is extremely complex due to its high computational complexity, and thus existing works solve the problem by designing approximation techniques or heuristics. 1) *Approximation solutions:* Considering the complexity of RSU deployment problem, an approximate solution can be derived by employing some mathematical tricks. For example, Zhang *et al.* transform the non-convex RSU deployment problem into a standard convex optimization problem by using tricks such as semidefinite relaxation to generate the solution [40]. However, when the problem scale becomes large, this approach is likely to be ineffective. 2) *Heuristic solutions:* Many of existing works rely on heuristic algorithms, including genetic algorithms [42], greedy algorithms [43], memetic algorithms [44], particle swarm algorithms [45], and more, to form the final solutions. These methods can obtain optimal or suboptimal solutions, but they also suffer from disadvantages such as the lack of effective iteration stop conditions, unstable performance, and poor scalability.

In contrast to above works, we model the general RSU deployment problem as a learning problem and use the reinforcement learning model [46], [47] to intelligently search the RSU deployment plan by leveraging rich features extracted from various input data. Moreover, previous works usually output a complete deployment plan, while *Greta* determines deployment sites sequentially, facilitating gradual RSU deployments to support emerging V2X services.

3 Background and Motivation

3.1 Trend of large-scale RSU deployment

Many automakers and governments are investing heavily in building vehicle infrastructure systems and developing V2X technologies to enable future fully autonomous driving [9]. For example, Audi completed the world's first open road test of L4 autonomous driving using V2X signals at the 2021 Wuxi Internet of Things (WIoT) Exposition [48]. Similarly, car companies such as Ford, Volkswagen, and more are gradually adopting V2X as a standard configuration for their new car products. In addition, many governments, such as China and the United States, have launched a series of initiatives to promote the wide adoption of V2X [9].

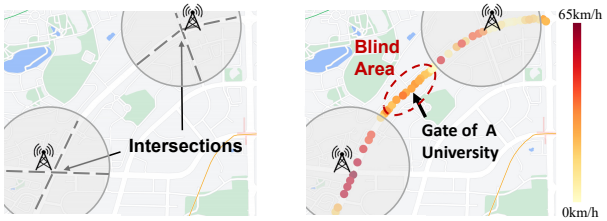


Fig. 1. (a) Typical RSU deployment at road intersections. (b) Mobility hints to RSU deployment by analyzing traffic data.

According to the latest market report, the global V2X market size is expected to grow from $2.6 billion in 2022 to $19.5 billion in 2028 [49].

As communication gateways, RSUs are core building blocks in the V2X infrastructure, and many pilot projects for RSU deployment have been initiated globally. For example, many cities in China have already deployed a considerable number of RSUs [50]. For example, Wuxi city has deployed over 400 RSUs, covering about 700 km of roads. Other cities including Chongqing, Guangzhou, Changsha, and Wuhan have also deployed 120, 130, 200, and 200 RSUs, respectively.

3.2 Inefficiency of existing deployment methods

In addition to industry and government efforts, there is also active research on designing effective RSU deployment strategies. Many of them take a static road map as input and output a set of road segments as the deployment sites to maximize the service coverage of available RSUs. To this end, they usually choose places with dense roads, such as road intersections, to deploy RSUs. Figure 1(a) shows typical RSU deployments at road intersections.

In addition to road maps, a wealth of sensing data about urban traffics now can be collected. They can capture people's mobility and traffic demands from different dimensions, which also provides useful hints for RSU deployments and should be adopted. For example, we analyze real traffic data collected from vehicles driving on the road segment, as shown in Figure 1(a), and visualize the average driving speed of vehicles in Figure 1(b). The vehicles pass the road intersections at high speed, while they instead have a long sojourn time in the middle of the road segment, which is a blind area in existing RSU deployments. By checking the road map, we find that there is a university gate, and thus drivers slow down to avoid accidents or temporarily stop to pick up or drop off passengers. Therefore, it is necessary to deploy RSUs in the middle of the road segment in Figure 1(b), so that RSUs can serve more vehicles and benefit traffic control. However, such deployment sites are difficult to infer from the static road network. Therefore, it is important to incorporate more data to extend the input dimensions for searching the best deployment sites.

On the other hand, rich V2X services will emerge, and they inevitably pose different requirements on RSU deployments. Existing methods, however, mainly take road coverage as the optimization target [13], [51], which may not be suitable for future V2X services. In fact, there exist many metrics that could be used to evaluate RSU deployments:

- *Road coverage* measures the number of road segments covered by the deployed RSUs.

- *Traffic coverage* reports the volume of vehicles that can be served by the deployed RSUs.
- *Communication quality indicators* investigate the quality-of-service (QoS) of communication provided by the deployed RSUs. The indicators include data rate, latency, throughput, connection time, and so on.

Given the enormous potential of V2X, more new metrics can be added to evaluate RSU deployments, e.g., vehicular mobility prediction accuracy, trajectory reconstruction accuracy, V2V-based vehicle coverage, and so on. In addition, RSUs will play multifaceted roles in the future to support various V2X services, e.g., serving as an edge computing server and communication gateway at the same time. Therefore, the consideration of RSU deployment is likely to be not limited to a single metric, but a weighted combination of multiple metrics, and the requirements for RSU deployment will therefore become more complex, diverse, and variable.

In summary, we find that existing RSU deployment methods mainly rely on the static road map to optimize individual metrics, and thus are inefficient to meet the requirements of emerging V2X services. To close the gap, we expect an RSU deployment framework that can analyze candidate deployment sites from different perspectives with rich input data and adapt to different optimization metrics.

3.3 Overview of the *Greta* design

In this paper, we present a general RSU deployment framework *Greta* to effectively deploy available RSUs. Figure 2 illustrates the architecture of *Greta*, which consists of three main modules: *input information library*, *reinforcement learning (RL)-based deployment model* and *output metric library*.

The input information library contains rich data related to RSU deployments, such as road maps, traffic data, POI information, and more. This library can be extended with new input data sources. Instead of using raw data directly, we devise a data profiling network to extract high-level features from raw data. These features are then adaptively fused and encoded as a comprehensive input representation.

Taking the fused features as the input state, the RL-based deployment model treats the RSU deployment problem as a learning problem and automatically explores the deployment environment to search for valuable deployment sites. Site search is guided implicitly by some reward functions that take into account the combination of multiple metrics in the output metric library. According to different requirements of RSU deployments, the RL model can intelligently adjust the weights between different input features and output metrics to produce the best deployment actions.

After an efficient exploration of the search space, *Greta* can generate a deployment strategy to maximize the total reward.

4 Design

4.1 Formulation of general RSU deployment problem

Different from previous works [12], [51], [52], [53], [54] that merely consider limited data input and a specific optimization target, in this paper, we consider a general RSU deployment problem, which can be expressed as follows: *given a set of candidate deployment sites* $\mathbb{L} = \{\ell_i | i = 1, \cdots, M\}$, *where* M *is the number of candidate sites, we aim to make full utilization of various available sensing data that are relevant with RSU deployments to determine a subset* $\mathbb{X} = \{x_j | j = 1, \cdots, N\} \subset \mathbb{L}$, *where* N *is the number of available RSUs subject to the budget, such that* $\mathbb{X}$ *can maximize a combination of weighted deployment targets.* Before formulating the problem, we first present the concepts of *input information library* and *output metric library*.

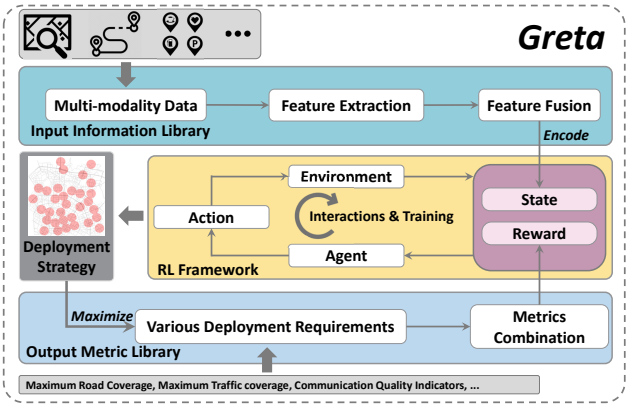


Fig. 2. The architecture of the *Greta* framework.

- **Input information library** $\mathcal{I}$ includes a variety of data sources, e.g., road map, traffic information, POI distribution, *etc.*, which may influence the RSU deployments. Each data source $i_i \in \mathcal{I}$ captures the demands on RSUs from different perspectives, and all of them together reflect the comprehensive RSU demands. Previous works usually consider the static road map only, while the input information library $\mathcal{I}$ contains diverse data sources and is also extensible for embracing new data to provide more accurate and effective guidance on the RSU deployments.
- **Output metric library** $\mathcal{O}$ incorporates various deployment metrics, each of which $o_j \in \mathcal{O}$ can be used to evaluate the effectiveness of an RSU deployment plan $\mathbb{X}$. Different V2X services potentially have distinct requirements on RSU deployments, which thus call for varied deployment metrics, such as road/traffic coverage or communication quality indicators. To meet the requirements of co-existing and emerging V2X services, a combination of multiple metrics are more preferred, and the weights among these metrics can be adaptively adjusted according to the RSU-supported V2X services.

Denote the weight for each output metric o_i as w_i, where $\sum_{i=1}^{K} w_i = 1$ and K is the total number of metrics in $\mathcal{O}$. If a metric is not required to be considered in the deployment, its weight is zero. Therefore, we can define the RSU deployment problem as follows:

$$\max_{\mathbb{X}} \quad \sum_{j=1}^{K} w_j \times o_j, \tag{1}$$

$$s.t. \quad f(\mathcal{I}, \mathbb{X}) = \{o_j\}, \text{ where } o_j \in \mathcal{O}, \tag{2}$$

$$|\mathbb{X}| \leq N, \text{ where } \mathbb{X} \subset \mathbb{L}, \tag{3}$$

where Eq. (1) aims to maximize the overall RSU deployment utility measured from various deployment metrics o_j, $f(\mathcal{I}, \mathbb{X})$ in Eq. (2) represents the impact on each output metric o_j given input sensing data $\mathcal{I}$ and deployment sites

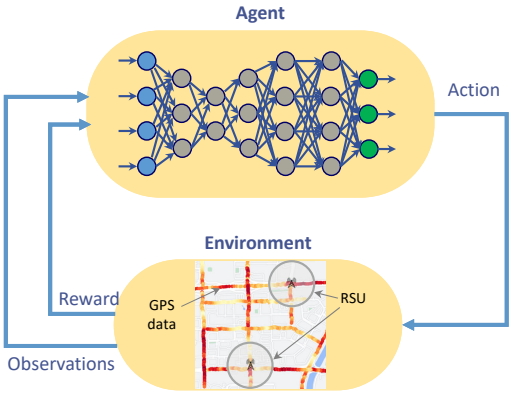


Fig. 3. The MDP-based RSU deployment process.

$\mathbb{X}$, and Eq. (3) ensures the number of RSUs does not exceed the budget N.

4.2 RSU deployments as a learning problem

However, it is non-trivial to directly solve the formulation in Eqs. (1)-(3) because function $f(\cdot)$ in Eq. (2) is difficult to obtain explicitly. In this paper, we propose to convert optimization to a learning problem to derive a practical solution. In particular, we exploit a multi-step decision learning to derive the deployment solution, where the best deployment sites in $\mathbb{X}$ are generated sequentially according to the available information in $\mathcal{I}$ and these already selected deployment sites. Moreover, multi-step decision learning can be well modeled by the Markov Decision Process (MDP) [55], which is a practical framework to solve decision-making problems in uncertain environments by defining a set of states, actions, and rewards. Therefore, we define these key elements for the RSU deployment problem as follows:

- **State** s encodes information about the deployment environment that can be described with the features extracted from various input data $\mathcal{I}$. In addition, these already deployed RSUs can be encoded to guide the deployment of additional RSUs.
- **Action** a selects one candidate site ℓ_i from $\mathbb{L}$ as the next RSU deployment site. In principle, all locations on the road network can be viewed as potential deployment sites, and the deployment granularity could be adjusted according to the requirements of V2X services.
- **Reward** r is the feedback of each applied action. In our problem, reward r can be the quantified influence of actions on the combined deployment metrics. In practice, we can either assign intermediate rewards to each action to generate dense rewards, or set rewards for each action to zero and only give the ultimate reward.
- **Police** π is the core of the MDP framework and defines the transition probability distribution among states. In our case, function $f(\cdot)$ is the expected policy π to guide the sequential RSU deployments.

Therefore, we model the RSU deployment problem as a sequential decision-making process, which caters to the trend of large-scale and gradual RSU deployments in practice. Moreover, the RL modeling can effectively reduce the

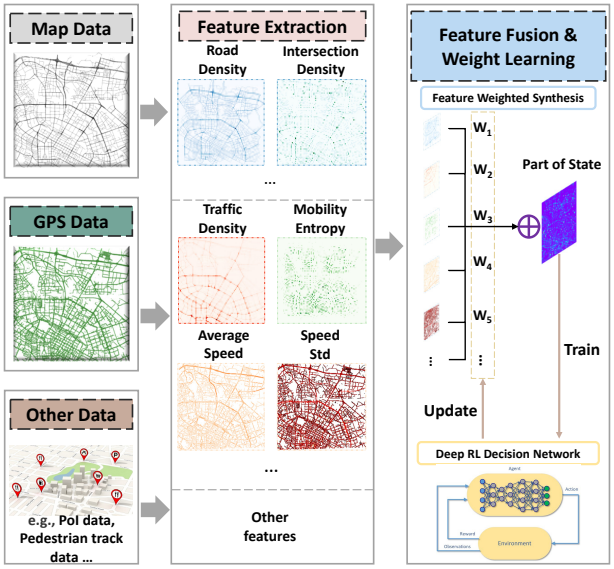


Fig. 4. Framework of data profiling network, which consumes various data sources from the input information library $\mathcal{I}$, and exploits the feature extraction layer and feature fusion layer to derive the environment representation.

computational complexity while incorporating rich urban data to derive better deployment solutions. The key to addressing such a problem is to find the suitable policy $f(\cdot)$ that can produce a reasonable action a given the input state s. An action will deploy one RSU on the road network and thus changes the deployment environment, which generates a new state that can be used to determine the next RSU deployment site, just as illustrated in Figure 3. Since candidate deployment sites $\mathbb{L}$ are known in advance, policy $f(\cdot)$ works like a classifier that categorizes different states and assigns a label, e.g., selected or unselected, to each candidate deployment site based on current input state. A large number of samples are required to train such a classifier, while collecting such data is difficult or even impossible due to the expensive cost of deploying RSUs.

Recognizing the above challenge, we exploit reinforcement learning (RL) [47], which is well suited to the MDP modeling, to solve the learning problem. More specifically, we employ the deep reinforcement learning (DRL) technique [46] to solve the general RSU deployment problem by directly learning the best deployment policy $f(\cdot)$. There are several advantages to adopting DRL to address the RSU deployment problem. First, DRL can accomplish challenging tasks by exploring and exploiting during the process of interacting with the deployment environment. As a result, it can get rid of the expensive collection of labeled samples.

Second, DRL is scalable and provides us adequate design space on the state representation, actions, and reward function. Specifically, we can incorporate diverse data sources in $\mathcal{I}$ into the states for comprehensively describing the deployment environment, and consider flexible combinations of various deployment metrics in $\mathcal{O}$ to define reward functions. Moreover, we only need to make appropriate modifications to the states, actions, and rewards to make the DRL solution adapt to other similar RSU deployment problems.

Third, DRL supports delayed rewards, which can help address the challenge of evaluating RSU deployment met-

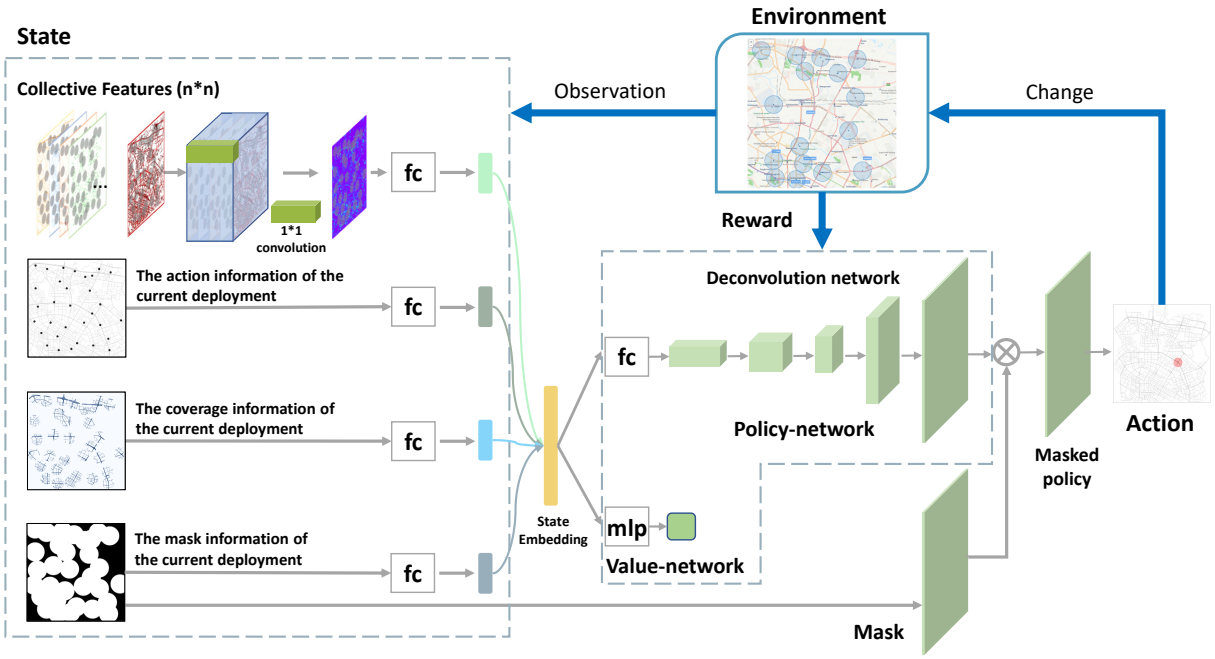


Fig. 5. Architecture of the DRL-based RSU deployment framework. Different input data sources are used to generate the state representation, which serves as the input of the policy network and the value network. The policy network outputs a probability distribution over the masked grids through a multi-layer deconvolution network, and the value network generates the expected reward for the current deployment through a simple MLP network.

rics in multi-step decision-making problems. When the complete deployment plan is not yet determined, it can be difficult to evaluate the effectiveness of intermediate deployment decisions. While delayed rewards allow the agent to focus on the overall objective and find a better solution, rather than being limited to short-term optimizations. Specifically, the reward for each action is set to zero, and the model is trained only on the final reward, which enables the agent to consider the long-term effects of its actions and make decisions that optimize the ultimate goal.

4.3 Collective features as environment representation

To comprehensively represent the deployment environment, *Greta* uses a data profiling network to automatically extract useful features from various input data available in $\mathcal{I}$ to form part of the state representation. As shown in Figure 4, the data profiling network consists of two layers, i.e., *feature extraction layer* and *feature fusion layer*.

Feature extraction. Although we have various data sources in the information library $\mathcal{I}$ to guide RSU deployments, these data are in different modalities with varied dimensions. To exploit such heterogeneous data, we partition the road network into $n \times n$ grids, and then extract grid-level features from each data source. As a result, the features derived from all data sources are in the uniform size of feature matrices, which facilitates feature fusion.

For each data source $i_i \in \mathcal{I}$, we first classify the data samples into different grids according to the samples' associated locations. For each grid, we extract some statistical feature that describes the local deployment environment for the RSU deployments. All grid-level features then form an $n \times n$ feature matrix. Noting that we may compute several feature matrices from one single data source. Different data sources indeed have distinct properties, while we preprocess them following a similar way, and thus the multi-modal input data sources can be well utilized by *Greta*. As concrete examples, we illustrate the common features extracted from some typical data sources as follows.

- *Road map* is the most important data source since we deploy RSUs at the roadside to serve the passing-by vehicles. For a given road map, we can derive the road density feature, which summarizes the road distribution, and the intersection density feature, which contains the statistic of road intersections for all grids.
- *Vehicular trajectory data* record the driving details, e.g., time-stamped location and speed, of vehicles. Such data directly reflect traffic flows and traffic conditions over the road network. Therefore, we can extract rich features from vehicular trajectory data, including traffic density feature, mobility entropy feature, average speed feature, and speed variations feature. In particular, to compute the mobility entropy feature, we count the vehicle turns at each intersection and calculate the probability of a vehicle turning to a certain road at a certain intersection accordingly. The derived mobility entropy can reflect the traffic complexity of each intersection.

Input information library $\mathcal{I}$ can accept new data sources, e.g., POI data. Since POIs are usually the destinations of many citizens' trips, POI data thus can implicitly reflect the nearby traffic flows. *Greta* will extract feature matrices from the new data source with the grid-based feature extraction process to enrich the state representation in the future.

Feature fusion. Different feature matrices will have unequal contributions to the RSU deployment decisions, we

thus use a neural network to adaptively adjust the weights among all feature matrices. Specifically, we adopt an 1×1 convolutional kernel to perform the feature fusion from the derived feature matrices. The fused feature matrix is treated as the representation of the deployment environment, which is part of the DRL model's input state. Thus, the weights of different feature matrices (i.e., the kernel's parameters) can be continuously adjusted by training the DRL model.

4.4 DRL-based RSU deployment framework

We reformulate the general RSU deployment problem as a learning problem, and propose a DRL model to accomplish the deployments of N RSUs over the road network by following the learned policy function $f(\cdot)$.

Architecture. Figure 5 illustrates the architecture of *Greta*'s DRL-based RSU deployment framework, which comprises a policy network and a value network. The input to both networks is the state representation, which is a concatenation of embeddings computed from various input data sources. The feature embeddings are weighted and fused to form a state representation through multiple fully-connected (FC) layers.

The policy network is a multi-layer deconvolution network that produces a 2-D probability distribution over the action space. The value network, on the other hand, is a simple multi-layer perception (MLP) network with two hidden layers, which is used to predict the estimated value of the expected reward for the current placement. The policy network is then optimized to maximize the expected reward as estimated by the value network. The interaction between the policy and value network forms the basis for many RL algorithms, such as actor-critic methods [56]. To enhance the learning efficiency, we also design a mask that filters out unnecessary or infeasible deployment sites before each action is sampled.

Next, we will materialize each key element of DRL modeling to address the general RSU deployment problem.

4.4.1 Contextual state

In addition to the fused features that are derived from the input information library, *Greta* also considers the already deployed RSUs and domain knowledge to generate the contextual states. As illustrated in the left part of Figure 5, *Greta* constructs the contextual state using the embeddings of the following information:

- *Collective features* are the most important guidance information for RSU deployments. In practice, *Greta* will reset the feature values as zeros for the grids that are covered by deployed RSUs after each action.
- *Deployed RSU map* specifies the action coordinate information of the already deployed RSUs.
- *Influence map* indicates the coverage information of the current RSU deployments. Specifically, we keep the grid value if the grid is covered by some RSUs and exclude the value if it is not covered by any RSU.
- *Mask* contains information about the prohibited grids, which are not suitable for deploying RSUs, based on current deployment status and domain knowledge. For example, in order to improve the convergence speed and reduce the overlap among RSUs, we prohibit further deployments within a certain range of the already deployed RSUs. In addition, the grids without roads are considered to be unnecessary to deploy RSUs.

Fig. 6. (a) An example road map matrix after QGIS processing; (b) The pruned action matrix by applying the filter mask.

In our implementation, we normalize each dimension of the fused features to optimize the training efficiency. Instead of directly combining this information, we compute the embedding for each kind of contextual information through a simple fully connected neural network with two hidden layers (128×128), which finally output a 32-dimensional embedding. The derived embeddings are then concatenated to form the contextual state.

4.4.2 Deployment action

Each candidate deployment site in $\mathbb{L}$ potentially becomes a possible action that indicates the location to deploy an RSU. Since we partition the road network into $n \times n$ grids, we generate the action space, i.e., $\mathbb{L}$, using these grids. Thus, the size of the action space is n^2. For any input contextual state s, the policy of *Greta*'s DRL model will output a probability distribution of the current deployment action over the grids. The action a for state s is subsequently sampled from this probability distribution.

The action space size will greatly affect the model training and performance. Too large action space will prolong the training process and reduce the efficiency, while too small action space seems to be meaningless, since a large grid may contain many road segments to deploy RSUs. The best grid setting should be comprehensively evaluated according to the road network and application requirements. Given the action space with size $n \times n$, we still propose a heuristic pruning method to accelerate the model training.

Since RSUs are typically deployed along with road facilities, such as traffic lights and cameras, we initialize a *filter mask* by leveraging the road network to exclude infeasible deployment sites. To that end, we perform a dot-wise product between the map matrix and the $n \times n$ action matrix, so as to exclude grids without roads from the action space. However, the roads in the graph are too fine and when multiplying the road map matrix with the action matrix, it is likely to miss those sites that are very close to the roads. To enhance the system's fault tolerance, we thus create buffers for the roads in the map using Quantum GIS (QGIS), which thickens and simplifies the road network. Figure 6(a) demonstrates a road network processed by QGIS, and the resulting pruned action space is shown in Figure 6(b).

The above operation initializes the filter mask of *Greta* by pruning infeasible actions based on the road network information. Later, the filter mask needs to be continuously updated according to the latest RSU deployment actions.

To avoid redundantly deploying RSUs, we hope that the newly deployed RSUs keep a certain distance from these already deployed RSUs. Assume that the service coverage radius of an RSU as r, we thus prohibit further deployments within the distance $\eta \times r$ of the already deployed RSUs, where η is a scaling parameter. We set $\eta \in [0, 2]$, and in particular $\eta = 2$ indicates no service coverage overlap between any two RSUs. Once a new RSU is deployed, we update the filter mask by setting the grids within $\eta \times r$ of the newly deployed RSU as infeasible deployment sites. Therefore, the domain knowledge enhanced filter mask can help *Greta* greatly reduce the action space, and thus improve the computation efficiency.

4.4.3 Reward

The objective of our DRL modeling is to maximize the long-term rewards that are used to approximate the requirements of V2X services. Specifically, *Greta* links the DRL's rewards with the output metric library $\mathcal{O}$. Before defining the reward function, we introduce the evaluation mechanisms for some fundamental metrics as follows:

- *Road coverage*: $\frac{\sum_{i=1}^{N} l(x_i)}{L}$, where $l(x_i)$ represents the road length covered by RSU x_i deployed by an action, and L is the total length of all roads in the road network.
- *Traffic coverage*: $\frac{\sum_{i=1}^{N} t(x_i)}{T}$, where $t(x_i)$ is the number of unique vehicles covered by RSU x_i, and T is the total number of vehicles observed within the road network.
- *RSU overlap*: $\frac{N*c-C}{N*c}$, where C is the actual area covered by all deployed RSUs, and c is the theoretical coverage area of each RSU, which can be set as the area of a circle, centered at the deployment site with coverage radius r.

In *Greta*, the reward function can be calculated using one or more metrics, with different weights assigned to each metric. Dense rewards can be generated by calculating an intermediate reward for each action based on the chosen metrics. However, some application requirements are highly dependent on the whole deployment plan, resulting in that intermediate rewards cannot be easily calculated for the required metrics. For example, if the deployment objective is to maximize communication quality, it is likely that the relative positions of all RSUs need to be considered in the formulation of the reward. In this case, it is not possible to give an intermediate reward for the deployment of each individual RSU. Instead, an ultimate reward needs to be given after all RSUs have been deployed.

When the deployment requirements are complex to evaluate, it becomes difficult or time-consuming to directly evaluate the ultimate reward. In such a case, it becomes necessary to approximate the reward. Since DRL training often requires numerous episodes to converge, the evaluation of approximated reward function should be fast.

We give an example of approximated reward design for the RSU-based mobility prediction application, which has been introduced in Section 6.2. The direct reward can be set as the prediction accuracy, while it requires both training of the prediction model given an RSU deployment scheme and testing to get the exact prediction accuracy. It is operationally difficult, and thus we can approximate the prediction accuracy by considering the prediction difficulty within the coverage of the deployed RSUs. This approximation is based on the intuition that a higher prediction difficulty leads to a lower prediction accuracy. As a result, measuring the prediction difficulty allows us to obtain a rough estimate of the prediction accuracy. Specifically, we define the prediction difficulty of each location y as $O(y)$, which can be customized. The sum of prediction difficulty within the area covered by RSUs is then used as a reward to measure prediction accuracy, that is:

$$R_{predict} = \sum_{y \in area} O(y).$$

Model training. Through the repetition of episodes (consisting of sequences of states, actions, and rewards), we train a policy π_θ modeled by a neural network that learns to take the best action for a particular state. The objective for the general RSU deployment problem is to maximize the expected reward over deployment strategies that are generated by the policy network. Parameters θ of the policy are trained using Proximal Policy Optimization (PPO) [57], which employs a clipped objective function. The core idea behind PPO is to restrict the policy update within a small range using a clip, and the objective function, referred as the clipped surrogate objective function, is given as:

$$L^{CLIP}(\theta) = \hat{E}_t[\min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)],$$

where $\hat{E}_t$ is the expected reward of parameters θ at timestep t, r_t represents the ratio between the new policy and the old one, and $\hat{A}_t$ is the estimated advantage.

Algorithm 1 PPO-clip for *Greta*

Input: Features extracted from $\mathcal{I}$
Initialize: Policy network θ, value network θ_v;
for epoch = 1,2,$\cdots$, *max_epochs* **do**
  // **Generate several RSU deployment strategies with current policy π_θ.**
  buffer.clear();
  for i=1,2,$\cdots$,*steps_per_epoch* **do**
    $\log p \leftarrow \pi(a|\theta)$;
    $\log p = mask(\log p)$
    $a \leftarrow \log p.sample()$;
    $v \leftarrow V(\theta_v)$;
    $s, r \leftarrow env.step(a)$;
    buffer.append$(a, \log p, v, s, r)$;
    if *TrajEnd()* **then**
      Save the best deployment $\mathbb{X}$ currently found;
      env.reset();
    end if
  end for
  Compute advantage estimates $\hat{A}_t \leftarrow buffer$ based on current $V(\theta_v)$;
  // **Update the policy by maximizing the PPO-Clip objective as shown in (1) with Adam.**
  $d\theta \leftarrow compute\_loss\_pi(buffer)$;
  Update θ;
  $d\theta_v \leftarrow compute\_loss\_v(buffer)$;
  Update θ_v;
end for
* Action: a, Value: v, State: s, Reward: r.
* *env*: RL environment
* *TrajEnd()*: All RSUs have been deployed.

Algorithm 1 presents the pseudocode of the model training. The algorithm first initializes the parameters of the policy and value networks. Then for each epoch, several RSU deployment strategies are generated by sampling from the probability distribution of the actions returned by the current policy. The generation of every RSU deployment strategy is termed as a *trajectory*. For each trajectory, *Greta* starts with an undeployed blank map canvas. It then generates the deployment plan by iteratively performing an action computed by the policy to the current map environment until the trajectory ends (line 6-16). The trajectory is terminated when the required number of RSUs is already deployed. At the end of each epoch, we compute the loss of policy gradient and critic gradient to update the parameters (line 19-22).

5 Performance Evaluation

5.1 Experiment setup

5.1.1 Implementation

We implement *Greta* in Chengdu city, China, using one month of GPS trajectory data that were collected from the contract vehicles of Didi Chuxing [19], a Chinese ride-hailing platform, in October 2016. The contract vehicles are required to upload their real-time status information every 3 seconds, which includes GPS location, travel speed, and direction. During the data collection phase, these vehicles totally generated more than 20 million GPS records per day. We download the road network of Chengdu city from OpenStreetMap [58]. Besides, we set the service coverage radius of each RSU as $r = 500$ meters.

The raw GPS trajectory data are pre-processed before in use as follows. First, the GPS records are grouped by vehicle ID and sorted by time to form logical and coherent trips. Next, we remove any trip that is too short because it may be incomplete or travel out of the testing area. In addition, we remove any stay points where a vehicle remains stationary for an extended period of time. Finally, we employ the fast map matching algorithm [59] to map the trajectories onto the road network, which can correct erroneous GPS locations and recover the vehicles' actual travel routes.

After data pre-processing, we divide the road network into $n \times n$ grids, and then extract features from the data. The visual representation of each feature is shown in Figure 4, similar to a heatmap. The resolution of the grids determines the amount of feature information and will affect the system performance. By default, we set $n = 84$ for the experiments.

We implement the PPO algorithm to train the DRL model using the SpinningUp framework [60], which is developed by OpenAI and can make use of GPUs to accelerate the training. The RL environment is implemented in Python to facilitate the use of SpinningUp. Table 1 presents the hyperparameters involved in the RL modeling.

TABLE 1
Hyperparameter settings for the RL modeling (The default setting of each hyperparameter is marked in bold).

Hyperparameters	**Value**
Max length per trajectory	{8, 16, **32**, 64, 128}
Max epochs	512
Max length per epoch	{512, 1024, **2048**, 4096, 8192}
Activation function	ReLU
MLP hidden size	**512x512**
Actor Learning rate	1.00E-04
Critic Learning rate	1.00E-3
Optimizer	Adam
PPO loss	
clipping ratio ϵ	0.2
Entropy coeff	0.01
GAE lambda λ	0.97
Discount factor γ	{0.75, 0.8, **0.93**, 0.965, 0.98}

5.1.2 Baselines

Because existing research works primarily focus on optimizing road/traffic coverage, we thus compare *Greta* with other baselines on these optimization objectives. Based on the literature review (see more in Section 2), we broadly categorize existing RSU deployment methods into three groups: *1) Naïve methods*; *2) Road information-based methods*; *3) Traffic data driven methods*. Based on this categorization, we select four representative RSU deployment methods from among them as the baseline methods for performance comparisons. These baseline methods are described as follows:

- *Uniform* [53], [61]: This naïve method will deploy RSUs uniformly on the road map without considering information of traffic flows or road network topology. Despite its simplicity, this method can provide uniform service coverage across the city.
- *I-RSU* [51], [62]: This is an intersection-based heuristic RSU deployment method with the goal of maximizing road coverage. *I-RSU* prefers to deploy RSUs at road intersections according to their density distribution. As a result, it can derive large road coverage.
- *CDA-DC* [52]: This is another representative intersection-based RSU deployment method that maximizes the traffic coverage by evaluating the centrality of each road intersection to determine its importance as a potential RSU deployment site. It deploys RSUs at intersections with higher importances.
- *Traffic-RSU* [29], [63], [64]: This method makes use of vehicular mobility data to guide the RSU deployments. Specifically, it divides a road map into grids and assigns traffic data into these grids, then deploys RSUs to certain grids with the goal of maximizing traffic coverage.

In addition, to verify the search performance of DRL in the RSU deployments, we also choose the Greedy Search (GS) algorithm and the Simulated Annealing (SA) algorithm as the baseline.

- *GS-RSU* [65]: GS-RSU also selects the RSU deployment site in a sequential process, while it selects the optimal deployment site, e.g., owning the largest value on the considered metric, at each time in a greedy manner, without considering the overall situation.
- *SA-RSU* [18]: SA is a powerful, yet slow, optimization method that can approximate global optimization in a large search space. Similar to RL, SA can optimize any non-differentiable cost function. In the implementation, we first select N random candidate deployment sites as the initial deployment plan. After that, we select any one of the initial N sites, and

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2023.3335853

Fig. 7. Comparisons on the RSU deployment plans of different methods with different available numbers of RSUs, i.e., $N = 8, 16, 32, 64, 128$. The upper three rows visualize the deployment results of three baselines, respectively, and the lower three rows show the results of *Greta*'s variants that aim to optimize road coverage, traffic coverage, and the weighted combination of road and traffic coverage, respectively.

switch it with any one of the remaining unselected sites. By comparing the change on some performance metrics, the algorithm then completes the update of one deployment site in the plan. The execution goes on iteratively, and whenever a better deployment solution emerges, the algorithm records it.

5.2 Evaluation results

In this section, we first comprehensively compare *Greta* with all other deployment methods (§5.2.1), and later we evaluate *Greta* from different aspects, including searching capability (§5.2.2), effectiveness of sequential deployments (§5.2.3), and computational complexity (§5.2.4). Lastly, we conduct sensitivity analysis by studying the impacts of different parameter settings on *Greta* (§5.2.5).

5.2.1 Comparisons with different deployment methods

To demonstrate the superiority of *Greta*'s adaptivity on different optimization targets, we modify the objective of *Greta* to produce three variant methods, i.e., *Greta-road*, *Greta-traffic*, and *Greta-weighted*, which aim to optimize the deployment target of road coverage, traffic coverage, and

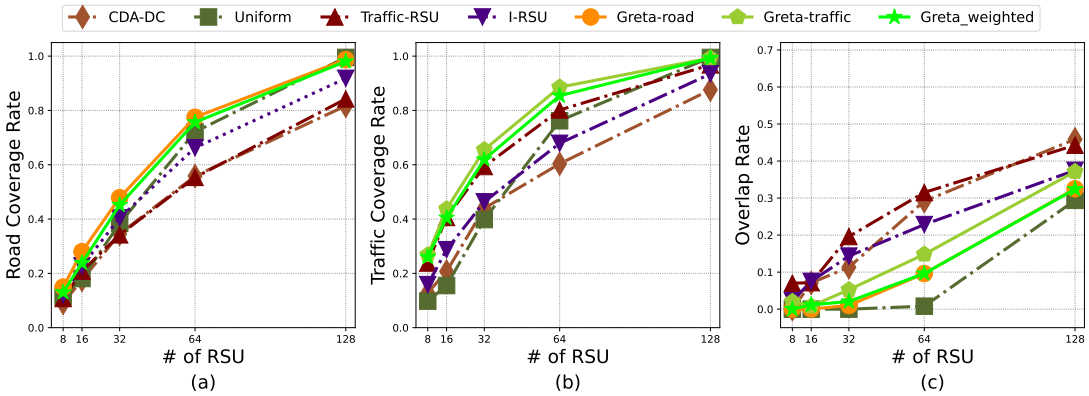


Fig. 8. Performance comparisons on (a) road coverage, (b) traffic coverage, and (c) overlap rate for different methods with various RSU budgets.

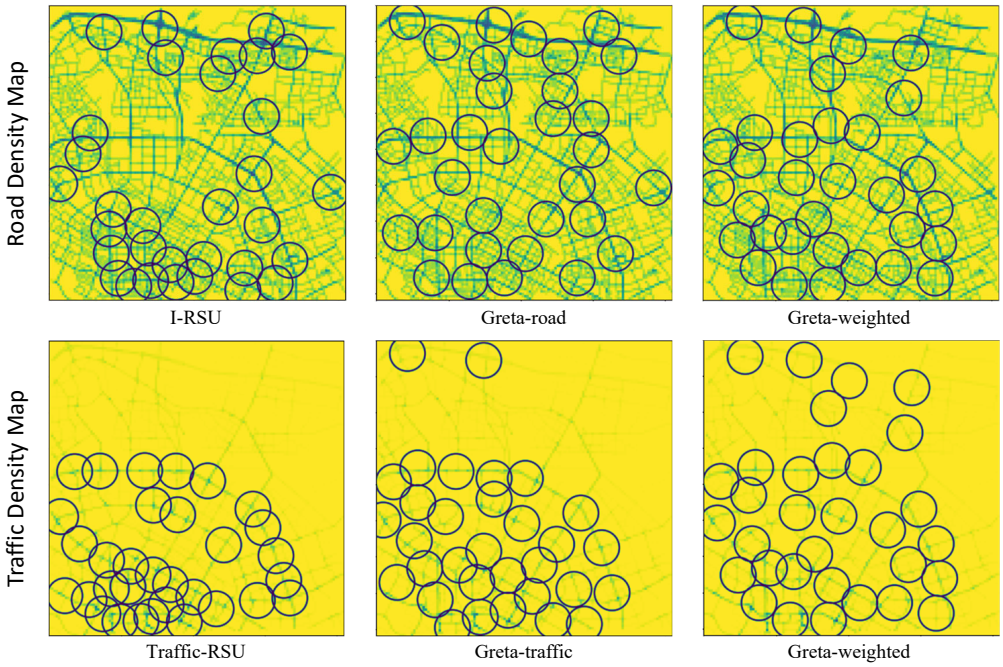


Fig. 9. Deployment results on feature map for different methods with $N = 32$ RSUs.

combined metrics of road and traffic coverage, respectively. These variant methods use the corresponding rewards as described in Section 4.4.

We visualize and compare the deployment results of different methods under various RSU budgets (i.e., $N = 8, 16, 32, 64, 128$) in Figure 7. From the figure, we see that these methods have distinct strategies to deploy RSUs, leading to different distributions of deployed RSUs. When the deployment problem is small-scale (i.e., small N), the deployments made by baseline methods may be reasonable under their respective deployment logic. However, once more RSUs need to be deployed, their deployment results become unreasonable and less effective. For example, the *CDA-DC* and *Traffic-RSU* have relatively high overlap rates when the number of RSUs is large (e.g., $N \geq 64$). In contrast, the proposed *Greta* performs well in different target settings, and can produce even distributed deployment plans as shown in Figure 7.

We summarize the performance of different methods under various performance metrics, and show the statistical results in Figure 8. With more RSUs to deploy, all methods can provide larger road coverage (see Figure 8(a)) or traffic coverage (see Figure 8 (b)). *Greta* brings relatively more improvement with 32 deployments. For the given road map with too small or too large number N of RSUs to deploy, the performance difference among different methods is not significant. Specifically, in the 32-RSU case, *Greta-road* (*Greta-traffic*) improves *Uniform*, *CDA-DC*, *I-RSU*, and *Traffic-RSU* on the road coverage (traffic coverage) by 24.8%, 38.6%, 18.5%, and 40.0% (64.6%, 49.0%, 41.4% and 10.4%), respectively. We also find that the variant *Greta-weighted*, which aims to optimize a combined metric of road and traffic coverage, can achieve similar performance as the variant that is particularly designed for a specific metric. For example, *Greta-weighted* almost derives the same road coverage as *Greta-road* as shown in Figure 8(a). It implies that *Greta* can automatically adjust the weights among multiple features to maximize the targeted requirement. In addition, Figure 8(c) shows the overlap rate of all deployed RSUs. In practice, we hope RSUs are evenly distributed, and thus smaller overlap rate is preferred. From Figure 8(c), we find that the three variants of *Greta* perform well and output reasonable deployment plans.

To intuitively understand the advantage of *Greta*, we

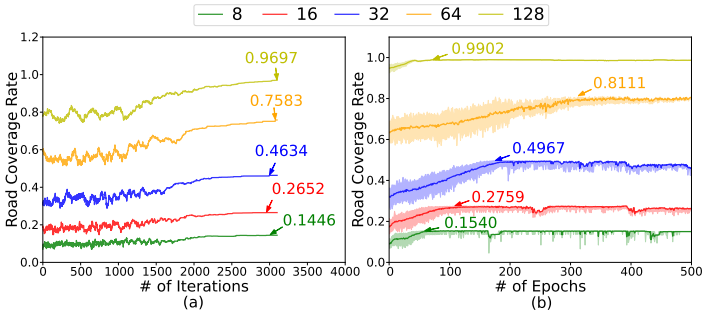


Fig. 10. Performance comparison between (a) SA-RSU and (b) *Greta*.

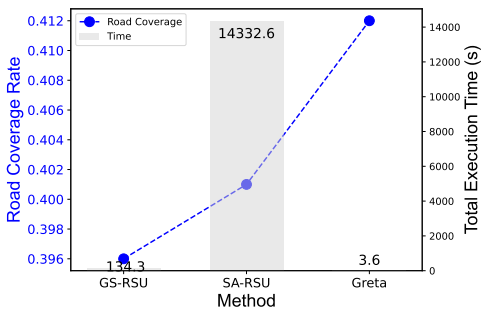


Fig. 11. Performance comparisons of sequential deployment.

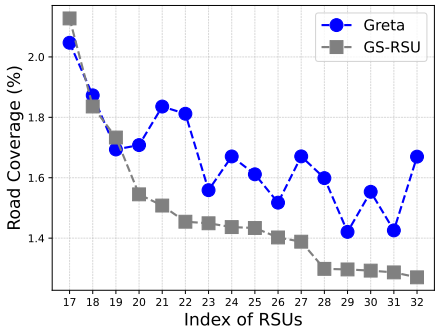


Fig. 12. Road coverage for each deployment site selected by *Greta* or *GS-RSU*.

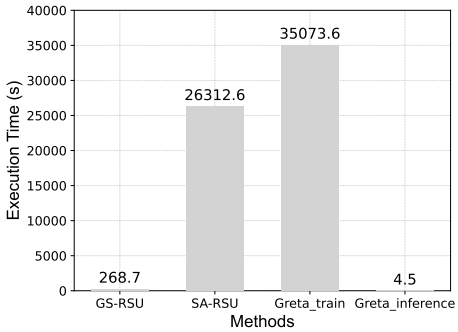


Fig. 13. Computational complexity comparisons of different methods.

visualize the deployment results of 32 RSUs on the corresponding feature maps (i.e., road density and traffic density) for different methods in Figure 9. For the road density (traffic density) feature, we see that *Greta-road* (*Greta-traffic*) can deploy RSUs to cover more feature-rich areas when compared to *I-RSU* (*Traffic-RSU*). In addition, we also visualize the deployment results of *Greta-weighted* and find it provides a satisfactory deployment plan that effectively balances the feature effects of road density and traffic density.

In summary, the results in Figures 7, 8, and 9 show that *Greta* outperforms the heuristic methods in large-scale deployments, thanks to the efficient exploitation of real-world data. Meanwhile, RL-based problem modeling makes *Greta* to be generalized for different deployment requirements.

5.2.2 Comparison with search-based method

To examine the solution space exploration ability of RL, we compare *Greta-road* with *SA-RSU* under the same problem setting that aims to maximize the road coverage for a given number N of RSUs. The greedy algorithm (*GS-RSU*), by its nature, simply selects the current optimums at each step and combines them to form the final solution, lacking the ability to explore the solution space. Therefore, we exclude *GS-RSU* from this experiment.

Figure 10 (a) and (b) show the training processes of *SA-RSU* and *Greta-road*, respectively, where the corresponding optimal values are given. By varying N from 8 to 128, *Greta-road* shows 6.5%, 4.0%, 7.2%, 7.0%, and 2.1% improvement than *SA-RSU* on road coverage, respectively.

Furthermore, *Greta* offers potential advantages in problem modeling compared to the SA algorithm. It allows for the flexibility of modifying the optimization objective, such as minimizing the required number of RSUs to achieve a targeted road coverage. For instance, if a road coverage requirement of 75% is specified, *Greta-road* can learn a deployment plan that utilizes the minimum number of RSUs. However, such a task cannot be accomplished by the *SA-RSU* method.

5.2.3 Effectiveness of sequential deployments

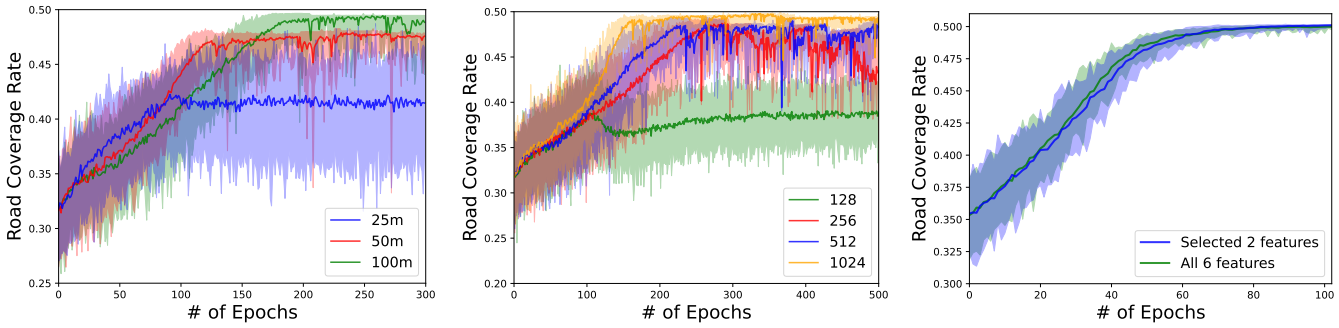
In this experiment, we consider a sequential incremental RSU deployment scenario. We firstly employ a uniform deployment strategy [53] to deploy 16 RSUs on the road network, and then separately use *GS-RSU*, *SA-RSU*, and *Greta* to deploy another 16 RSUs atop these deployed RSUs.

Figure 11 presents the performance comparison results on road coverage and execution time. We find that although both *GS-RSU* and *Greta* are sequential deployments, *Greta* can go beyond the local optimum because it chooses deployment sites from a global view and thus obtains more globally favorable deployment actions at each time. Furthermore, since *Greta* and *GS-RSU* can quickly generate the next deployment action based on the current deployment situation without re-training or re-searching, we thus provide a detailed comparison of each deployment action of these two methods in Figure 12. Although *GS-RSU* can compute the optimal deployment sites as *Greta* for the initial three RSUs, it cannot always get the best sites in the latter process, as the road coverage of *GS-RSU*'s selected sites is much smaller than the ones provided by *Greta*.

Unlike *Greta* and *GS-RSU*, *SA-RSU* can only generate the complete deployment plan by re-searching the locations of the remaining 16 deployment actions from scratch, within the constraints of having 16 deployment actions already in place, so it takes a much longer execution time as shown in Figure 11. Because *SA-RSU* does not produce intermediate deployment actions, it is excluded from the detailed comparisons in Figure 12

5.2.4 Evaluation of computational complexity

As *Greta* is built on reinforcement learning, its computational efficiency is related to various factors, including



(a) Action resolutions (b) Hidden sizes (c) Feature numbers

Fig. 14. Impacts of different factors on *Greta*.

the size of state space and action space, the adopted RL algorithm, and among others. It is difficult to mathematically analyze *Greta*'s computational complexity, and thus we conduct experiments to compare *Greta* with two baseline methods, i.e., *GS-RSU* and *SA-RSU*, on the time complexity of training and inference.

Figure 13 shows the experimental results. *GS-RSU* takes 268.7 seconds for each deployment plan search, while *SA-RSU* needs a total of 26312.6 seconds to generate the complete deployment plan. Although *Greta* takes a relatively long time for training, its online inference is extremely fast, i.e., 4.5 seconds, once the model has been well trained. Moreover, the trained model can be used for additional deployments in the future, while *SA-RSU* still takes a long time to generate a new complete plan.

5.2.5 Sensitivity analysis

To gain further insights into the effects of different factors on *Greta*, we conduct a series of sensitivity analyses on *Greta* by running *Greta-road* with $N = 32$ RSUs.

- *Impact of action resolution*: The action resolution is determined by the grid sizes. We compare the performance of *Greta-road* with varied grid sizes, i.e., 25, 50, and 100 meters. From the results shown in Figure 14(a), we observe that higher action resolution (i.e., smaller grid size) leads to faster convergence, while the derived road coverage becomes worse. In particular, when the action resolution is 25 meters, the training fails to converge with drastic fluctuation until the end. Therefore, it is important to carefully select the action resolution to balance convergence speed and final performance when applying *Greta* in practical applications.
- *Impact of hidden layer size*: We investigate the impact of the hidden layer size in the MLP network, which is used by the value network, by varying it from 64×64 to 512×512. The results shown in Figure 14(b) reveal that the MLP networks with different hidden layer sizes converge to similar results, except for the network setting with 128, and the difference among these settings lies in the convergence speed. This is because a more complex network structure can model intricate relationships better.
- *Effect of feature fusion*: *Greta* can fuse features and adaptively adjust their weights according to the application's requirement. To verify its effectiveness, we only take the two features with larger weights after feature fusion as the input for *Greta-road*. Compared with the variant using all six available features as shown in Figure 4, the results in Figure 14(c) show that both variants achieve the same performance. It proves that *Greta* indeed can identify the useful features and assign them with larger weights.

6 Discussion

In this section, we will discuss the limitations and potential applications of *Greta*.

6.1 Limitations and open issues

Despite the huge advantages, we also realize some limitations of *Greta* in its current implementation. We discuss these limitations and hope to inspire future research efforts.

- **Privacy protection.** Due to the advantages of adapting to different RSU deployment requirements, *Greta* can well support various V2X applications, e.g., mobility prediction and trajectory reconstruction. These applications involve the vehicles' location information and thus may arise concerns about user privacy. To enable such smart mobility applications while protecting users' privacy, we may adopt an anonymous data-sharing mechanism by allowing vehicles to upload anonymous data to RSUs, as these applications only require vehicular trajectory data rather than personal information.
- **Extension to mobile RSUs.** In the current *Greta* design, we only consider deploying stationary RSUs on the road network, while some recent works [66], [67] propose to deploy RSUs on moving vehicles as mobile RSUs. By providing occasional service for vehicles out of the coverage of stationary RSUs, mobile RSUs can effectively enlarge the service coverage of all RSUs. However, how to jointly optimize the deployments of stationary and mobile RSUs, while still considering different requirements of V2X services, is an interesting yet challenging research problem that we plan to address in our future work.
- **Better RSU communication model.** The majority of RSU deployment works model the RSU communication range as either 2D circular or 1D linear, which facilitates the problem formulation but may

not fully capture the urban environments. Obstacles such as buildings can significantly attenuate communication signals and affect the accuracy of prediction models. We thus believe that if we can construct a more comprehensive RSU communication model, either theoretically or through some data-driven approaches, by considering the influence of the urban environments, the derived RSU deployments would be more effective and efficient.

6.2 Potential applications of *Greta*

Due to its superiority in adapting to dynamical RSU deployment requirements, *Greta* can well support various V2X applications. Here we list some potential applications.

- **Mobility prediction:** Mobility modeling is essential for understanding people's travel habits [68] and enabling various mobility services [69], [70]. RSUs can serve as sensors to observe vehicles' movements within a city, and thus a potential application is to predict the location of a specific vehicle, even if it is out of RSUs' sensing coverage. *Greta* can optimize RSU deployment to reduce mobility uncertainty by exploiting features extracted from various input data, such as mobility entropy and traffic volume. By covering intersections with higher mobility entropy and more traffics, where vehicle turning operations are hard to predict, the uncertainty on a vehicle's location is reduced, and the mobility prediction accuracy can be improved.
- **Trajectory reconstruction:** Complete trajectory data are useful for many trajectory mining applications, e.g., transit system optimization, infrastructure planning, POI recommendations, traffic sensing and monitoring, etc [71]. From sparse RSU observations, trajectory reconstruction application aims to recover the route a vehicle actually traveled on. As an offline task, this application can take advantage of global RSUs' information for recovering a vehicle's actual travel route. Therefore, by deploying RSUs to maximize both road and traffic coverage, *Greta* can potentially improve the reconstruction accuracy.
- **V2V communication optimization:** As one of the key functionalities, RSUs provide communication service for vehicles and can also serve as the relay nodes between vehicles for information exchange. In such a case, RSUs supplement the communication gap to improve V2V communication quality [53]. To well support such an application, *Greta* can be employed to deploy RSUs that meet the communication requirements, such as vehicle connectivity, communication delay, *etc.*

7 Conclusion

In this paper, we present *Greta*, a general RSU deployment framework that aims to improve existing methods with better design utility and deployment scalability. To achieve this goal, *Greta* incorporates an input information library and an output metric library, both of which are adjustable and extensible to consider rich sensing data or new/updated deployment requirements related to RSU deployments. In addition, *Greta* exploits reinforcement learning (RL) to model the general RSU deployment problem as a learning process, and customize the RL model to automatically explore the deployment environment to find good deployment strategies. A prototype system of *Greta* is implemented and experimentally evaluated using real-world data. The results demonstrate the effectiveness of *Greta*. Compared to existing RSU deployment methods, *Greta* can achieve great performance gains on various metrics.

Acknowledgments

This work was supported in part by the National Key Research and Development Project (2019YFB2102300), the China NSFC Grant (No.61936014 and No.62172284), the Shanghai Municipal Science and Technology Major Project (2021SHZDZX0100), the Shanghai Science and Technology Innovation Action Plan Project (No. 22511105300), the Fundamental Research Funds for the Central Universities, the grant of Guangdong Basic and Applied Basic Research Foundation (No.2022A1515010155), and the CityU strategic research grant (No. 7005658).

References

[1] H. Bagheri, M. Noor-A-Rahim, Z. Liu, H. Lee, D. Pesch, K. Moessner, and P. Xiao, "5g nr-v2x: Toward connected and cooperative autonomous driving," *IEEE Communications Standards Magazine*, vol. 5, no. 1, pp. 48–54, 2021.

[2] A. Ghosal and M. Conti, "Security issues and challenges in v2x: A survey," *Computer Networks*, vol. 169, p. 107093, 2020.

[3] D. Zhang, "Distributed cache enabled v2x networks: Proposals, research trends and challenging issues," *arXiv preprint arXiv:1803.06059*, 2018.

[4] I. Soto, M. Calderon, O. Amador, and M. Urueña, "A survey on road safety and traffic efficiency vehicular applications based on c-v2x technologies," *Vehicular Communications*, vol. 33, p. 100428, 2022.

[5] C. Lochert, B. Scheuermann, C. Wewetzer, A. Luebke, and M. Mauve, "Data aggregation and roadside unit placement for a vanet traffic information system," in *Proceedings of the fifth ACM international workshop on VehiculAr Inter-NETworking*, 2008, pp. 58–65.

[6] C.-H. Ou, "A roadside unit-based localization scheme for vehicular ad hoc networks," *International Journal of Communication Systems*, vol. 27, no. 1, pp. 135–150, 2014.

[7] E. R. Magsino, "Employing mobility traces' findings in deploying roadside units in an urban setup," in *2019 IEEE 9th Symposium on Computer Applications & Industrial Electronics (ISCAIE)*. IEEE, 2019, pp. 97–102.

[8] ASTRI. Astri's cellular vehicle-to-everything (c-v2x) technology. [Online]. Available: https://www.astri.org/smart-mobility/technology.html

[9] T. I. W. Exhibition, "V2x for cooperative transportation," https://www.toyota.co.jp/its/en/2021/v2x_deployment/, accessed January 4, 2023.

[10] J. Wright, J. K. Garrett, C. J. Hill, G. D. Krueger, J. H. Evans, S. Andrews, C. K. Wilson, R. Rajbhandari, B. Burkhard, *et al.*, "National connected vehicle field infrastructure footprint analysis," United States. Department of Transportation. Intelligent Transportation . . . , Tech. Rep., 2014.

[11] A. Guerna, S. Bitam, and C. T. Calafate, "Roadside unit deployment in internet of vehicles systems: a survey," *Sensors*, vol. 22, no. 9, p. 3190, 2022.

[12] S. Mokhtari, G. Mirjalily, C. M. Silva, J. F. M. Sarubbi, and J. M. S. Nogueira, "The deployment of roadside units in vehicular networks based on the v2i connection duration," in *2020 16th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, 2020, pp. 1–6.

[13] A. Guerna and S. Bitam, "Gica: An evolutionary strategy for roadside units deployment in vehicular networks," in *2019 International Conference on Networking and Advanced Systems (ICNAS)*. IEEE, 2019, pp. 1–6.

[14] X. Cao, Q. Cui, S. Zhang, X. Jiang, and N. Wang, "Optimization deployment of roadside units with mobile vehicle data analytics," in *2018 24th Asia-Pacific Conference on Communications (APCC)*. IEEE, 2018, pp. 358–363.

[15] G. Wang, J. Wu, T. Xu, and B. Tian, "3d vehicle detection with rsu lidar for autonomous mine," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 1, pp. 344–355, 2021.

[16] H. Liu, Y. Zhang, and T. Yang, "Blockchain-enabled security in electric vehicles cloud and edge computing," *IEEE Network*, vol. 32, no. 3, pp. 78–83, 2018.

[17] H. Xu, P. Zhou, R. Tan, M. Li, and G. Shen, "Limu-bert: Unleashing the potential of unlabeled data for imu sensing applications," in *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, 2021, pp. 220–233.

[18] A. Mirhoseini, A. Goldie, M. Yazgan, J. W. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, A. Nazi, *et al.*, "A graph placement methodology for fast chip design," *Nature*, vol. 594, no. 7862, pp. 207–212, 2021.

[19] "Didi gaia initiative," https://outreach.didichuxing.com/research/opendata/, accessed August 2022.

[20] A. Abdrabou and W. Zhuang, "Probabilistic delay control and road side unit placement for vehicular ad hoc networks with disrupted connectivity," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 1, pp. 129–139, 2011.

[21] T.-J. Wu, W. Liao, and C.-J. Chang, "A cost-effective strategy for road-side unit placement in vehicular networks," *IEEE Transactions on Communications*, vol. 60, no. 8, pp. 2295–2303, 2012.

[22] J. Tao, L. Zhu, X. Wang, J. He, and Y. Liu, "Rsu deployment scheme with power control for highway message propagation in vanets," in *2014 IEEE Global Communications Conference*, 2014, pp. 169–174.

[23] Y. Wang, J. Zheng, and N. Mitton, "Delivery delay analysis for roadside unit deployment in vehicular ad hoc networks with intermittent connectivity," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 10, pp. 8591–8602, 2016.

[24] M. Rios, V. Marianov, and M. Pérez, "Locating fixed roadside units in a bus transport network for maximum communications probability," *Transportation Research Part C: Emerging Technologies*, vol. 53, pp. 35–47, 2015.

[25] Z. Zhang, C. Li, L. Yu, Y. Zhao, and Y. Li, "A multi-objective roadside units deployment method in vanet," in *2021 IEEE International Conference on Smart Internet of Things (SmartIoT)*. IEEE, 2021, pp. 390–394.

[26] F. Yang, C. Zhao, X. Ding, and J. Han, "An analytical model for energy harvest road side units deployment with dynamic service radius in vehicular ad-hoc networks," *IEEE Access*, vol. 8, pp. 122 589–122 598, 2020.

[27] D. Kim, Y. Velasco, W. Wang, R. Uma, R. Hussain, and S. Lee, "A new comprehensive rsu installation strategy for cost-efficient vanet deployment," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 5, pp. 4200–4211, 2016.

[28] E. R. Magsino and I. W.-H. Ho, "An enhanced information sharing roadside unit allocation scheme for vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 15 462–15 475, 2022.

[29] H. Yu, R. Liu, Z. Li, Y. Ren, and H. Jiang, "An rsu deployment strategy based on traffic demand in vehicular ad hoc networks (vanets)," *IEEE Internet of Things Journal*, vol. 9, no. 9, pp. 6496–6505, 2021.

[30] S. Mehar, S. M. Senouci, A. Kies, and M. M. Zoulikha, "An optimized roadside units (rsu) placement for delay-sensitive applications in vehicular networks," in *2015 12th Annual IEEE consumer communications and networking conference (CCNC)*. IEEE, 2015, pp. 121–127.

[31] B. Cao, S. Fan, J. Zhao, S. Tian, Z. Zheng, Y. Yan, and P. Yang, "Large-scale many-objective deployment optimization of edge servers," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3841–3849, 2021.

[32] Y. Li, D. Jin, P. Hui, and S. Chen, "Contact-aware data replication in roadside unit aided vehicular delay tolerant networks," *IEEE Transactions on Mobile Computing*, vol. 15, no. 2, pp. 306–321, 2016.

[33] S. Liu, B. Guo, K. Ma, Z. Yu, and J. Du, "Adaspring: Context-adaptive and runtime-evolutionary deep model compression for mobile applications," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 5, no. 1, pp. 1–22, 2021.

[34] L. Jiang, T. G. Molnár, and G. Orosz, "On the deployment of v2x roadside units for traffic prediction," *Transportation Research Part C: Emerging Technologies*, vol. 129, p. 103238, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0968090X21002515

[35] S. Ghosh, I. Saha Misra, and T. Chakraborty, "Optimal rsu deployment using complex network analysis for traffic prediction in vanet," *Peer-to-Peer Networking and Applications*, vol. 16, no. 2, pp. 1135–1154, 2023.

[36] Y. Liang, H. Liu, and D. Rajan, "Optimal placement and configuration of roadside units in vehicular networks," in *2012 IEEE 75th Vehicular Technology Conference (VTC Spring)*. IEEE, 2012, pp. 1–6.

[37] C. Park, H. Ko, and Y. Kyung, "Cost-efficient edge cloud deployment method for autonomous driving," in *2022 13th International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2022, pp. 789–792.

[38] A. Moubayed, A. Shami, P. Heidari, A. Larabi, and R. Brunner, "Cost-optimal v2x service placement in distributed cloud/edge environment," in *2020 16th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, 2020, pp. 1–6.

[39] Y. Liang, S. Zhang, and Y. Wang, "Data-driven road side unit location optimization for connected-autonomous-vehicle-based intersection control," *Transportation Research Part C: Emerging Technologies*, vol. 128, p. 103169, 2021.

[40] B. Zhang, G. Zhu, S. Xu, and N. Zhang, "Energy-efficient roadside units deployment in vehicular ad hoc networks," in *6th International Conference on Wireless, Mobile and Multi-Media (ICWMMN 2015)*, 2015, pp. 6–10.

[41] Z. Gao, D. Chen, S. Cai, and H.-C. Wu, "Optimal and greedy algorithms for the one-dimensional rsu deployment problem with new model," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 8, pp. 7643–7657, 2018.

[42] Z. Li, J. Liu, and Q. Zhang, "A ga-based strategy for deploying cable connected roadside units in vanets," in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*. IEEE, 2018, pp. 1–5.

[43] R. Cai, Y. Feng, D. He, Y. Xu, Y. Zhang, and W. Xie, "A combined cable-connected rsu and uav-assisted rsu deployment strategy in v2i communication," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[44] S. Anbalagan, A. K. Bashir, G. Raja, P. Dhanasekaran, G. Vijayaraghavan, U. Tariq, and M. Guizani, "Machine-learning-based efficient and secure rsu placement mechanism for software-defined-iov," *IEEE Internet of Things Journal*, vol. 8, no. 18, pp. 13 950–13 957, 2021.

[45] M. M. Islam, M. M. Saad, M. T. R. Khan, and S. H. A. Shah, "Proactive uavs placement in vanets," in *ICC 2022-IEEE International Conference on Communications*. IEEE, 2022, pp. 1–7.

[46] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.

[47] M. A. Wiering and M. Van Otterlo, "Reinforcement learning," *Adaptation, learning, and optimization*, vol. 12, no. 3, p. 729, 2012.

[48] M. Wilkes, "Audi realizes the world's first open road test of l4 autonomous driving with v2x signaling at the 2021 wuxi wiot exposition," https://www.audichina.cn/cn/brand/en/news/2021/2021-10-24.html, accessed January 4, 2023.

[49] "Automotive v2x market," https://www.marketsandmarkets.com/Market-Reports/automotive-vehicle-to-everything-v2x-market-90013236.html, accessed January 4, 2023.

[50] "Status of large-scale deployment of v2x infrastructure in key cities in china," https://www.auto-testing.net/news/show-116699.html, accessed January 4, 2023.

[51] M. Mao, P. Yi, Z. Zhang, L. Wang, and J. Pei, "Roadside unit deployment mechanism based on node popularity," *Mobile Information Systems*, vol. 2021, 2021.

[52] Z. Wang, J. Zheng, Y. Wu, and N. Mitton, "A centrality-based rsu deployment approach for vehicular ad hoc networks," in *2017 IEEE International Conference on Communications (ICC)*. IEEE, 2017, pp. 1–5.

[53] J. Barrachina, P. Garrido, M. Fogue, F. J. Martinez, J.-C. Cano, C. T. Calafate, and P. Manzoni, "Road side unit deployment: A density-based approach," *IEEE Intelligent Transportation Systems Magazine*, vol. 5, no. 3, pp. 30–39, 2013.

[54] R. Shahin, A. A. El-Moursy, S. M. Saif, H. M. Abbas, and S. M. Nassar, "Fog node optimum placement and configuration technique for vanets," in *2020 International Conference on Communications, Signal Processing, and their Applications (ICCSPA)*. IEEE, 2021, pp. 1–6.

[55] M. van Otterlo and M. Wiering, *Reinforcement Learning and Markov Decision Processes*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 3–42. [Online]. Available: https://doi.org/10.1007/978-3-642-27645-3_1

[56] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," *Advances in neural information processing systems*, vol. 12, 1999.

[57] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[58] M. Haklay and P. Weber, "Openstreetmap: User-generated street maps," *IEEE Pervasive computing*, vol. 7, no. 4, pp. 12–18, 2008.

[59] C. Yang and G. Gidofalvi, "Fast map matching, an algorithm integrating hidden markov model with precomputation," *International Journal of Geographical Information Science*, vol. 32, no. 3, pp. 547–570, 2018. [Online]. Available: https://doi.org/10.1080/13658816.2017.1400548

[60] J. Achiam, "Spinning Up in Deep Reinforcement Learning," 2018.

[61] F. Zhan, P. Jing, and B. Ran, "Infrastructure allocation for improving sensing accuracy and connectivity probability based on combination strategy in vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 15 244–15 255, 2022.

[62] M. Lehsaini, N. Gaouar, and T. Nebbou, "Efficient deployment of roadside units in vehicular networks using optimization methods," *International Journal of Communication Systems*, vol. 35, no. 14, p. e5265, 2022.

[63] G. Premsankar, B. Ghaddar, M. Di Francesco, and R. Verago, "Efficient placement of edge computing devices for vehicular applications in smart cities," in *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2018, pp. 1–9.

[64] C. M. Silva, W. Meira, and J. F. Sarubbi, "Non-intrusive planning the roadside infrastructure for vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 938–947, 2015.

[65] T. Deng, L. Zhong, Y. Liu, and F. Lin, "A trajectory analysis-based critical intersection discovery approach for roadside unit deployment in vehicular networks," in *2022 11th International Conference on Information Communication and Applications (ICICA)*, 2022, pp. 63–68.

[66] J. Heo, B. Kang, J. M. Yang, J. Paek, and S. Bahk, "Performance-cost tradeoff of using mobile roadside units for v2x communication," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 9, pp. 9049–9059, 2019.

[67] T. Karunathilake and A. Förster, "A survey on mobile road side units in vanets," *Vehicles*, vol. 4, no. 2, pp. 482–500, 2022.

[68] L. Pappalardo, S. Rinzivillo, Z. Qu, D. Pedreschi, and F. Giannotti, "Understanding the patterns of car travel," *The European Physical Journal Special Topics*, vol. 215, pp. 61–73, 2013.

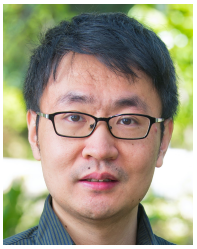
[69] M. Ye, L. Guan, and M. Quddus, "Mpbrp-mobility prediction based routing protocol in vanets," in *2019 International Conference on Advanced Communication Technologies and Networking (CommNet)*. IEEE, 2019, pp. 1–7.

[70] Y. Yang, X. Xie, Z. Fang, F. Zhang, Y. Wang, and D. Zhang, "Vemo: Enabling transparent vehicular mobility modeling at individual levels with full penetration," in *The 25th Annual International Conference on Mobile Computing and Networking*, 2019, pp. 1–16.

[71] C. Celes, A. Boukerche, and A. A. Loureiro, "Mobility trace analysis for intelligent vehicular networks: Methods, models, and applications," *ACM Computing Surveys (CSUR)*, vol. 54, no. 3, pp. 1–38, 2021.

Xianjing Wu received the B.E. degree in Electronic Information Science and Technology from the North University of China, Taiyuan, China, in 2018. He is currently pursuing the Ph.D. degree in Computer Science under the joint PhD programme of Tongji University, Shanghai, China and City University of Hong Kong, Hong Kong, China. His research interests include intelligent transportation, mobile computing and deep learning.

Zhidan Liu received the Ph.D. degree in computer science and technology from Zhejiang University, Hangzhou, China, in 2014. After that, he worked as a Research Fellow in Nanyang Technological University, Singapore. He is currently an Associate Professor with College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. His research interests include mobile computing, big data analytics, Internet of Things, and urban computing. He is a member of IEEE, ACM, and CCF.



Zhenjiang Li received the B.E. degree from Xi'an Jiaotong University, China, in 2007, and the M.Phil. and Ph.D. degrees from the Hong Kong University of Science and Technology, Hong Kong, China, in 2009 and 2012, respectively. He is currently an Associate Professor with the Department of Computer Science, City University of Hong Kong. His research interests include Internet of Things, edge AI systems and smart sensing.

Shengjie Zhao received the Ph.D. degree in electrical and computer engineering from Texas A&M University, TX, USA. He is currently the Dean of the College of Software Engineering, Tongji University, Shanghai, China. His research interests include artificial intelligence, big data, wireless communications, image processing, and signal processing. He is a senior member of IEEE.