2025 International Conference on Intelligent Computing
July 26-29, Ningbo, China
https://www.ic-icc.cn/2025/index.php

Joint Deployment Optimization of Fixed and Vehicle-Mounted Edge Servers for Urban Internet of Vehicles

Xuyang Chen[1&][0009-0004-5079-9866], Zhihai Tang[2&] [0009-0002-2937-5394], Jingtong Chen[1][0009-0000-8870-2602], Wei Song[1][0009-0006-8868-3134], Aiwen Huang[1][0009-0009-5760-2081], Le Chang[1*][0000-0002-1205-4220] and Heng Li[3][0000-0001-5592-7004]

[1] Guangdong University of Technology, Guangzhou, China
{3222001037, 3122000872, sonwe, 3121001002} @mail2.gdut.edu.cn
{lechang@gdut.edu.cn}
[2] Southeast University, Nanjing, China
{230249453@seu.edu.cn}
[3] Central South University, Changsha, China.
{liheng@csu.edu.cn}

Abstract. In the Internet of Vehicles (IoV), the user computation demand varies spatially and temporally. Thus, traditional static edge servers with fixed capacity at fixed sites lack the flexibility to handle such user dynamic. To this end, we study the joint deployment optimization of fixed and vehicle-mounted edge servers for an IoV system, where fixed servers (FESes) offer the basic coverage of the computation offloading service, and vehicle-mounted edge servers (VESes) focus on serving demand hotspots on the move. We first design the GICUNet traffic flow prediction model to precisely forecast the future traffic. Next, we allocate the computation capacity to each FES using Bayesian Optimization to minimize the deployment cost. We then design a Mobile Server scheduling algorithm based on Bipartite Graph Rematching (MS-BGR) to plan the short-distance paths of the VESes that cover most of the user demand. Experimental results show that our solution is excellent in terms of traffic prediction accuracy, adaptability to spatio-temporal dynamic user demand, and energy-efficiency of the VES travel paths compared with existing popular algorithms.

Keywords: Internet of Vehicles (IoV), Edge Computing, Vehicle-mounted Edge Serves, Traffic Prediction, Deployment, Path planning

1 Introduction

In recent years, the fast development of autonomous driving and the Internet of Vehicles (IoV) have spurred a variety of innovative applications, such as High-Definition (HD) map-assisted driving, cooperative lane change, and other advanced driver assistance systems [1]. These computing-intensive and delay-sensitive applications require

[&]The two authors contribute equally to this work.
[*] Corresponding author

considerable computing resources, which can be hardly accommodated on vehicles [2-4]. Therefore, the edge computing paradigm has become the promising solution which pushes the computing resources to the vicinity of these vehicles, i.e., *IoV nodes*, and serves them through computation offloading [5-8].

In edge computing-assisted IoV, edge servers can be placed at fixed sites such as cellular base stations, Roadside Units (RSUs), WiFi access points, street lamps etc., known as *fixed servers*, which provide the fundamental coverage of edge computing services. The servers can also be mounted on movable platforms such as trucks, buses, Unmanned Autonomous Vehicles (UAVs) etc., known as *mobile servers*, which enhance the computation elasticity to adapt to the spatiotemporally varying user demand [9].

The placement of the edge servers has attracted great attention, with several key challenges that need further exploration. One major challenge arises from the high-speed movement of the IoV nodes, leading to the drastic spatiotemporal variation of the computation workload of the edge servers. It is difficult to serve such varying workload even using mobile edge servers, as they take time to reach the hotspots [10]. Therefore, it is necessary to predict the locations of future hotspots to guide the scheduling of the mobile edge servers. Besides, the coordination between fixed and mobile servers is also an open problem. A rational allocation of the computing resources among fixed and mobile servers, together with an efficient path planning algorithm of the mobile servers should be carefully designed that can quickly respond to the dynamic traffic change and thus the computation workload variation of the system. However, most of the existing works treat the two categories separately. They either focus on the collaboration among base stations or mobile edge servers [11-13]. Only a few studies have considered the collaboration between fixed and mobile edge servers, but they ignore the energy consumption of mobile edge servers [14], or rely on complex algorithms such as Deep Reinforcement Learning (DRL) without prior knowledge [15].

To address the challenges above, we study the joint deployment optimization of fixed and vehicle-mounted edge servers for an IoV system, including the determining the computing capacity of the fixed servers (FESes) and planning the paths of the vehicle-mounted edge servers (VESes), with the objective of minimizing the total deployment cost and travel distance of the VESes. Considering that VESes are usually not fast enough to reach demand hotspots, we also design a UNet-based prediction model to predict future traffic that allows the VESes to move in advance. Our major contributions are summarized as follows.

- We build a joint-optimization model of the fixed and vehicle-mounted servers including capacity allocation and path planning, which minimizes the total deployment cost and travel distance of the servers.
- We design a traffic prediction model to accurately forecast the traffic volume (i,e., the computation workload) in different areas of a city in the near future, which serves as the basis for determining the capacity of the FESes and dispatching VESes.
- Based on the prediction result, we determine the capacity of fixed edge servers for each grid using Bayesian Optimization, and devise a path planning scheme for

scheduling the vehicle-mounted edge servers to serve the remaining workload hotspots.

The remainder of this paper is organized as follows. The related work is reviewed in Section 2. In Section 3, we present the joint-optimization model of the fixed and vehicle-mounted servers. In Section 4 we describe our proposed solution to the optimization problem. Section 5 shows and analyzes the experimental results. Section 6 concludes the paper and discusses the future work.

2 Related Works

Many existing works focus on the optimization strategies of already-deployed edge servers. Saurez et al. proposed the OneEdge architecture for dynamic scheduling and allocation of application resources, which involved monitoring the load of edge nodes and the end-to-end latency of applications through a central controller [16]. Alqe et al. employed reinforcement learning to train a resource management module to dynamically allocate the edge computing resources, optimizing user experience quality and ensuring task success rates [17]. Our previous works also studied the management of fixed-site servers, including placing them at proper locations, the allocation of the computation capacity, and the energy optimization of the VMs [18, 19].

However, existing static edge servers are unable to satisfy the rapidly changing demand of the IoV nodes. Therefore, dynamic configuration of the edge computing resources has become a hot research topic. Yang et al. constructed a Mobile Edge Computing (MEC) system assisted by multiple Unmanned Aerial Vehicles (UAVs) which achieved load balancing of UAVs and improved the efficiency of UAV mission execution [20]. Mishra et al. investigated the integrated synergistic effects between 5G/B5G cellular systems and UAVs [21]. They treated the UAVs as a new type of aerial User Equipment (UE) and integrated them into the existing cellular networks to improve the performance. Liu et al. studied UAV-Edge-Cloud computing systems, and the results showed that their approach achieved highly efficient and stable performance in an online UAV swarm environment regarding computation offloading and traffic routing [22]. Our previous works studied the placement of UAVs for urban IoV systems and designed efficient online heuristic algorithms to schedule the UAVs to cover the workload hotspots [23].

Considering the limitations of UAVs, such as limited battery endurance, payload capacity etc., researchers have proposed using service vehicles, e.g., trucks, to carry edge servers. Feng et al. proposed a vehicle-assisted offloading scheme that aimed to reduce the latency of computing tasks [24]. Liu et al. introduced a distributed vehicle-edge computing solution called Autonomous Vehicle Edge in the fields of vehicular networking and edge computing [25], which significantly improved the system utility and quality of service. Liu et al. proposed a multi-resource orchestration framework in VEC with an asynchronous deep reinforcement learning algorithm, which achieved low latency, high reliability and efficient utilization of resources for task processing [26]. Our previous work designed a vehicle-mounted edge server deployment scheme based on Adelson-Velsky and Landis (AVL)-tree, Gaussian Mixture Model and Simulated

Annealing, which demonstrated the superiority in terms of computation resource utilization and travelling distance [27]. We also proposed using buses to carry edge servers and provided a Dung Beetle-based computation offloading strategy using bus-mounted edge servers, which demonstrated good performance in terms of total cost and task success rate [28].

Our approach in this paper differs from the above works by combining fixed edge servers and vehicle-mounted edge server together to fulfill the needs of IoV nodes. We build a fine-grained mathematical model and design a 2-layer allocation strategy which harnesses fixed servers and mobile servers to maximize the edge computing performance with minimum deployment cost.

3 System Model and Problem Formulation

This section presents our system model of joint deployment optimization with both fixed edge servers and vehicle-mounted edge servers in an urban IoV system. The former ones offer basic coverage of the computation offloading service, and the latter ones add computation elasticity to the system to handle the spatio-temporal user dynamics.

3.1 System Model and Assumptions

We model an IoV system as a 2D map with $I \times J$ grids, where each grid represents an independent geographical area. The set of indices of these grids is denoted as $\mathcal{A} = \{1, 2, \ldots M\}$, where M represents the total number of grids. We deploy the following components within each grid shown in Fig. 1. User equipment (UE), i.e., IoV nodes carrying user devices, generate computing tasks periodically and offload these tasks to selected edge servers. The Fixed Edge Server (FES) is the fixed-site server with fixed computation capacity and stable energy supply. It receives, executes the computing tasks from the UE, and transmits the result back to the UE. The workload of a FES can be highly unstable due to the mobility of vehicles under urban environments. Therefore, we introduce Vehicle-mounted Edge Servers (VESes) to add elastic computing resource to handle such user dynamics. These VESes can be dispatched to different grids to compensate for the computing deficit there-in. The Base Station (BS) offers high-bandwidth network connections for all the components and the Task Scheduler collects the relevant information and assist the assignment of the computing tasks to the servers.

We assume there is no bottleneck on the network links within each grid, which allows us to focus on the scheduling of computation resources. Each UE (vehicle) generates time-sensitive computing tasks in a grid. These tasks must be executed and completed before the vehicle leaves the grid, through offloading to either the FES or VESes. We assume homogeneous offloading tasks of all UE nodes, and the total number of tasks in a grid is in proportion to the number of the UE nodes there-in. We adopt discrete time slots where the duration of each single time slot is Δt. A time slot is denoted by $t = 1, 2, \ldots, T$ where T is the total number of time slots, or namely the total time duration of our interests.

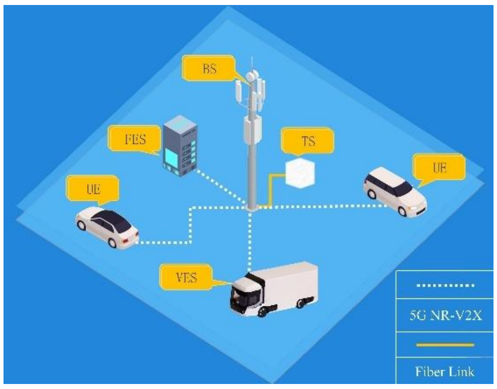

Fig. 1. Components within One Grid of the Edge Computing IoV System.

3.2 Joint Deployment of FESes and VESes

Let $w_m(t)$ denote the number of computing tasks in grid m during time slot t, i.e., the user demand. c_m^{FES} represents the computation capacity of the FES deployed at grid m, evaluated as the number of computing tasks that can be processed in one time slot. There are a total number of K VESes. We use a series of locations $l_{k,m}^{\text{VES}}(t)$ to represent their trajectories as follows.

$$l_{k,m}^{\text{VES}}(t) = \begin{cases} 1, & \text{VES k is in grid m at time slot t;} \\ 0, & otherwise. \end{cases} \tag{1}$$

The first constraint is the maximum travel distance that a VES can reach between two consecutive time slots, $D_{\max}$, i.e.,

$$\forall t \in [1, T), k, \sum_{i=1}^{M} \sum_{j=1}^{M} d_{i,j} \cdot l_{k,i}^{\text{VES}}(t) \cdot l_{k,j}^{\text{VES}}(t+1) \le D_{\max} \tag{2}$$

where $d_{i,j}$ denotes the distance between grid i and j.

We consider the energy endurance of the VESes. Let $e_k^{\text{VES}}(t)$ denote the energy of VES k at the beginning of time slot t, evaluated as the number of computing tasks that can be processed. The energy capacity of each VES is $e_{\max}^{\text{VES}}$. A VES can be recharged/refueled at any grid at any time slot if the energy is exhausted or no computing tasks are received, i.e.,

$$\text{if } e_k^{\text{VES}}(t) = 0\text{: } e_k^{\text{VES}}(t+1) = e_{\max}^{\text{VES}}; \tag{3}$$

$$\text{if } e_k^{\text{VES}}(t-1) = e_k^{\text{VES}}(t) \text{ and } l_{k,m}^{\text{VES}}(t-1) = l_{k,m}^{\text{VES}}(t) : e_k^{\text{VES}}(t+1) = e_{\max}^{\text{VES}} \tag{4}$$

The cost of our concern in this study includes the deployment cost of all the FESes and VESes, as well as the maintenance cost of the VESes. The deployment cost is evaluated as the total cost on the computation capacity of these servers:

$$C^{\text{FES}} + C^{\text{VES}} = \sum_{m=1}^{M} \left(p^{FES} \cdot c_m^{\text{FES}} + \sum_{k=1}^{K} l_{k,m}^{\text{VES}}(t) \cdot p^{VES} \cdot e_{\max}^{\text{VES}} \right) \tag{5}$$

where p^{FES} and p^{VES} are the unit deployment costs of the computation capacity of the FES and the VESes, respectively.

The maintenance cost of the VESes is evaluated as the total travel distance of all the VESes during the entire time duration from time slot 1 to T, denoted as D^{VES} and calculated as

$$D^{\text{VES}} = \sum_{k=1}^{K} \sum_{t=0}^{T} \sum_{i=1}^{M} \sum_{j=1}^{M} d_{i,j} \cdot l_{k,i}^{\text{VES}}(t) \cdot l_{k,j}^{\text{VES}}(t+1), \tag{6}$$

Let α and β denote the weights of the deployment cost and maintenance cost of the servers ($\alpha + \beta = 1$), respectively. The optimization problem is thus formulated as

$$\boldsymbol{Min}: \alpha\left(C^{\text{FES}} + C^{\text{VES}}\right) + \beta \cdot D^{\text{VES}} \tag{7}$$

$$s.t.\ l_{k,m}^{\text{VES}}(t) \in \{0, 1\}; \tag{8}$$

$$\forall t \in [1, T], k:$$

$$\sum_{i=1}^{M} \sum_{j=1}^{M} d_{i,j} \cdot l_{k,i}^{\text{VES}}(t) \cdot l_{k,j}^{\text{VES}}(t+1) \le D_{\max}; \tag{9}$$

$$\forall t \in [1, T-1], k:$$

$$e_k^{\text{VES}}(t+1) = e_k^{\text{VES}}(t) - \min\left(e_k^{\text{VES}}(t), w_m(t) - c_m^{\text{FES}}\right); \tag{10}$$

$$\text{if } e_k^{\text{VES}}(t) = 0: e_k^{\text{VES}}(t+1) = e_{\max}^{\text{VES}}; \tag{11}$$

$$\forall t \in [1, T-2], k:$$

$$\text{if } e_k^{\text{VES}}(t) = e_k^{\text{VES}}(t+1) \text{ and } l_{k,m}^{\text{VES}}(t) = l_{k,m}^{\text{VES}}(t+1): e_k^{\text{VES}}(t+2) = e_{\max}^{\text{VES}}; \tag{12}$$

$$\forall t \in [1, T], k: \sum_{m=1}^{M} l_{k,m}^{\text{VES}}(t) = 1; \tag{13}$$

$$\forall m: c_m^{\text{FES}} \ge 0; \tag{14}$$

where $l_{k,m}^{\text{VES}}(t) - s$ and $c_m^{\text{FES}} - s$ are the unknown variables. Constraint 8 means $l_{k,m}^{\text{VES}}(t)$ is {0, 1} variable. Constraint 9 ensures that the total distance traveled by each VES between consecutive time slots does not exceed the maximum travel distance,

$D_{\max}$. Constraint 10 updates the available energy of VES k of the next time slot based on its energy of the current time slot. Constraint 12 and 13 mean that if the energy of VES k is exhausted or if VES k is static and idle in one time slot, the energy of VES k will be replenished in the next time slot. Constraint 14 ensures that each VES is only deployed to one grid at any time slot. Constraint 15 specifies that the computation capacity of any FES is non-negative. This problem is NP-hard.

4 Heuristic Strategies

In this section, we present our heuristic strategies for the joint deployment problem with fixed and vehicle-mounted edge servers, i.e., to tackle the intractable NP-hard problem in the previous section.

4.1 Methodology

Firstly, we employ the deep learning technique to forecast the time-varying traffic of each grid, which is used to approximate the computation demand. Next, we utilize the Bayesian Optimization to analyze the traffic pattern of the grids and identify heavily-loaded grids and time slots. Based on such statistical results, we set appropriate computation capacity of the FES in each grid to accommodate most of the demand in regular hours, and dispatch VESes to those overloaded grids to handle the extra offloading requests that exceed the capacity of the FESes during peak hours. At last, we design a path planning algorithm to manage the transition of the VESes among the grids between different time slots, while minimizing the total travel distance and thus the energy consumption.

4.2 Forecast the Grid Traffic

We predict the traffic using a convolutional neural network model called UNet. Convolution were originally used for image processing, but in recent years, they have also been applied to traffic prediction [29]. Since adjacent grids are strongly correlated, better prediction performance can be achieved by capturing traffic features in the spatial dimension through convolution. UNet is structured through a combination of multiple convolutional modules, including upsampling and downsampling layers. In its original form, UNet employs standard convolutions, normalization layers, and activation functions in these modules. Given that the coordinate of grid m is(m_x, m_y), and recall that the demand in the grid at time slot t is $w_m(t)$, the predicted demand at time slot $t+1$ is calculated through the standard convolution as:

$$w^{\text{pre}}_{m_x, m_y}(c', t+1) = \sum_{i=-s}^{s} \sum_{j=-s}^{s} w_{m_x+i, m_y+j}(c, t) \cdot F_c(c, c', i, j) . \tag{15}$$

The output $w^{\text{pre}}_{m_x, m_y}(c', t+1)$ denotes the prediction result of the demand volume, the input $w_{m_x+i, m_y+j}(c, t)$ means the traffic data of its surrounding grids, c and c' stands for the channel of the features (traffic inflow and outflow in this paper) at time slot t and the predicted feature in t+1, respectively. The convolution kernel F_c extracts the features across the entire feature map.

However, standard convolutions are not ideally suited for our specific traffic prediction task, as they process all input channel c uniformly. To this end, we develop a unique convolutional module named GroupInceptionConvolution (GICConv). Specifically, GICConv comprises of two specialized convolutional layers: PointWise Convolutional (PWC) layer and Group Convolutional (GC) layer. The PWC layer, with a kernel size of 1, only establishes connections for the features within each grid. The GC layer employs a convolution kernel of size 3 and divides c into g groups, which means that the information about the surrounding grids will be captured, and spatial connections will be made for each group of features. Our feature extraction module is represented as

$$W^{\text{pre}} = \text{PWC}(\text{GC}(W)) + \text{GC}(\text{PWC}(W)), \tag{16}$$

where W and W^{pre} are the input and output data. By combining the two channels: PWC followed by GC, and GC followed by PWC, it is possible to capture global feature patterns while preserving internal feature relationships. The complete network architecture is based on UNet, which we refer to as GICUNet, as is shown in Fig. 2.

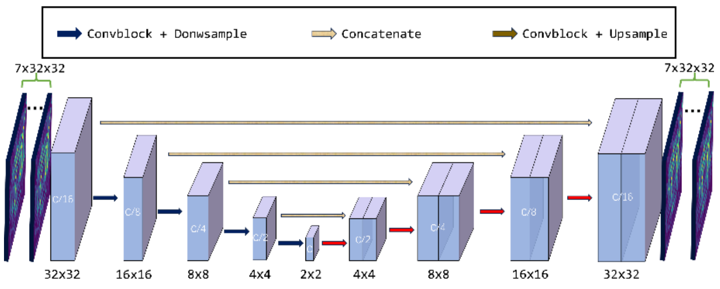

Fig. 2. The UNet-based model for Grid Traffic Prediction.

4.3 Determine the Computation Capacity of the FESes

Once the computation capacity of a FES is determined and deployed, it cannot be changed. We use Bayesian optimization (BO) to determine such computation capacity of each FES, with the aim of maximizing the server utilization and minimizing the cost, which is detailed in Alg. 1.

BO is an algorithm based on a probabilistic model that integrates exploration and exploitation. During the optimization process, BO considers both exploring unknown regions and exploiting known regions. Compared with grid search, random search or genetic algorithms, BO usually converges to the optimal solution with fewer evaluation iterations, which makes it especially suitable for global search and optimization problems in high-dimensional spaces.

For a given grid *m*, the prediction results for a specific period $[t_i, \ldots t_j]$ (e.g., 10 days) are extracted. The computation capacity of the FES at this grid can be set to any value between the maximum and minimum demand within this period, which forms the exploration space for BO. The working process is shown in the Fig. 3. The blue line with dots represents the user demand in the time domain. The exploration range of BO, i.e., the computation capacity of the FES, is between the two red lines. If the computation capacity finally converges to the green dashed line as the optimal solution, the sum of the cost on the orange part and the blue part should be minimized.

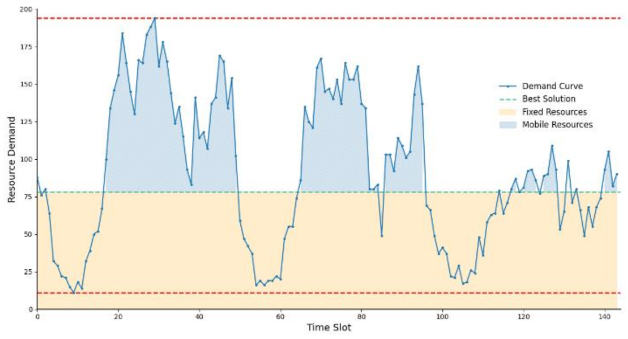

Fig. 3. Illustration of BO.

Recall that the unit deployment cost of the computation capacity of the FES is p^{FES}, and the unit deployment cost of the VES is p^{VES}. Given a proportion ρ and set the computation capacity of the FES to $c_m^{FES} = \rho \cdot max_{t \in [t_i, t_j]} w_m(t)$. The objective is to minimize the total cost on computation capacity, which can be formulated as:

$$\text{Min:} \sum_{t=t_i}^{t_j} \left(p^{\text{FES}} c_m^{\text{FES}} + p^{\text{VES}} \max\left(w_m(t) - c_m^{\text{FES}}, 0 \right) \right) . \tag{17}$$

To solve the problem, the BO algorithm will first select several initial points within the exploration space and calculate the spending. It then uses Gaussian Process to build a probability model based on the observed results. Next, acquisition functions will be used to determine the next point to evaluate, and the new result will be used to update the Gaussian process model. The process will be repeated until convergence or the preset number of iterations is reached.

4.4 Plan the Paths of VESes

After determining the capacity of the FESes, we will dispatch VESes to provide complementary services during the surge periods. The goal is to minimize the energy consumption of the VESes, i.e., the total travel distance, while meeting user demands. We

Algorithm 1 Bayesian Optimization for Optimal Fixed Resource Allocation per Grid

```
Require: Resource demand matrix $w_m(t)$, for each grid cell $m_x = 1, \ldots, 32$,
$m_y = 1, \ldots, 32$, and each time slot $t = t_i, \ldots, t_j$; Fixed Resource cost $p^{\mathrm{FES}}$, Mobile Resource cost $p^{\mathrm{VES}}$, number of time slots $n = t_j - t_i$;
Ensure: Capacity matrix $c^{\mathrm{FES}}_{m_x, m_y}$ and cost matrix $E_{m_x, m_y}$ for all grid cells.
1: for $m_x = 1$ to 32 do
2:   for $m_y = 1$ to 32 do
3:     Let $L \leftarrow \{ w_m(t) \mid t = 1, \ldots, n \}$
4:     Compute $l_{\min} \leftarrow \min(L)$ and $l_{\max} \leftarrow \max(L)$
5:     if $l_{\min} = l_{\max}$ then
6:       Set optimal fixed resource $x^* \leftarrow l_{\min}$
7:       Compute cost $e \leftarrow p^{\mathrm{FES}} \cdot x^* \cdot n$
8:     else
9:       Define the objective function:
         $$f(x) = x \cdot p^{\mathrm{FES}} \cdot n + p^{\mathrm{VES}} \sum_{t=t_i}^{t_j} \max\{0,\ w_m(t) - x\}$$
10:      Apply Bayesian Optimization (using the TPE method) on $x \in [l_{\min}, l_{\max}]$ to obtain:
         $$x^* = arg\ min_{x \in [l_{\min}, l_{\max}]} f(x)$$
11:      Set cost $e \leftarrow f(x^*)$
12:    end if
13:    Update matrices: set $c^{\mathrm{FES}}_{m_x, m_y} \leftarrow x^*$ and $E_{m_x, m_y} \leftarrow e$
14:  end for
15: end for
16: return Capacity matrix $c^{\mathrm{FES}}_{m_x, m_y}$ and cost matrix $E_{m_x, m_y}$
```

design a Mobile Server scheduling algorithm based on Bipartite Graph Rematching (MS-BGR) to achieve that.

Our MS-BGR is an iterative matching-based path planning strategy which considers the energy endurance of the VESes and aims at satisfying the user demand with minimum energy consumption. At the end of each time slot, the energy status of all the VESes will be reviewed. The VESes with available energy will be sent to different grids according to the time varying-demand of the next time slot, and exhausted VESes will halt for one time slot to get recharged/refueled. Our objective is to match the VESes with the demanding grids such that the VESes can support a maximum volume of the user demand with their remaining energy with minimum total travel distance. Fig. 4 shows one iteration of matching for a given time slot. The left part of the bipartite graph represents the grids with user demand, and the right part represents all the VESes with remaining energy. An edge connecting grid i and VES j means that the VES is able to move to grid i within the travel distance limit $D_{\max}$ from its current residing grid. $w'_i(t)$ represents the current demand volume in grid i at time slot t. The weight of each edge $W_{i,j}$ should reflect the preference to reduce the travel distance of the VESes, so we define it as

$$W_{i,j}=\gamma_1\times\min(e_j^{\mathrm{VES}},w'_i(t))-\gamma_2\times d_{i,j}. \tag{18}$$

The first term means that we prefer to send a VES to a grid with a small difference of the supply and demand, which reduces the number of moving VESes globally. The second term aims to minimize the total travel distance. γ_1 and γ_2 indicate the preference of the two terms, respectively. Thus the problem is to find the maximum weight matching in the bipartite graph, which can be solved using the Hungarian Algorithm (Kuhn-Munkres Algorithm) in polynomial time. This process is repeated until no more matching can be found for the time slot. The algorithm is detailed in Alg. 2.

Although MS-BGR currently operates on predicted demand, its iterative bipartite matching structure inherently supports real-time adjustments. By feeding live traffic updates into the weight matrix (Eq.18), the algorithm can dynamically rematch VESes to emerging hotspots within the travel distance constraint. This online adaptation capability will be formalized in future work using model predictive control frameworks.

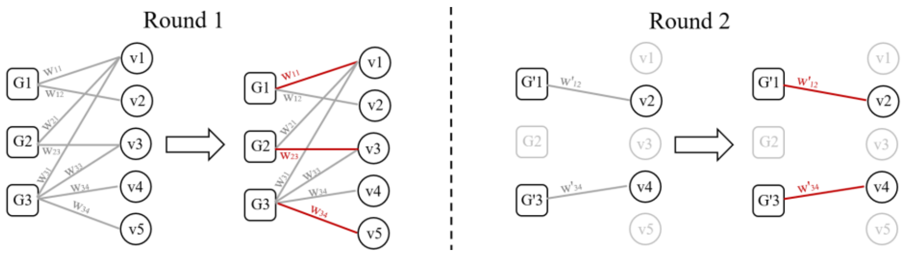

Fig. 4. Process of MS-BGR.

5 Performance Evaluation

5.1 Experiment Setup

1) **Data Set**: We use the TaxiBJ dataset released in [30]. In the dataset, the urban area of Beijing is divided into 32×32 areas, and the number of vehicles entering and exiting each area for each half an hour is recorded from July 1, 2013 to October 30, 2013, from March 1, 2014 to June 30, 2014, from March 1, 2015 to June 30, 2015 and from November 1, 2015 to April 10, 2016.

2) **Data Preprocessing**: We take advantage of the 7-day user behavior cycle, and use the traffic volume of the time slots in the previous 7 days to predict the traffic volume of the same time slots in the following 7 days. Subsequently, we randomly split the data into an 80% portion for model training and the remaining 20% portion for testing.

```
Algorithm 2 Path Planning Algorithm of VES
Require: Grids as left nodes $\mathscr{N} = \{1,2,...,n\}$, mobile servers as right nodes
$\mathscr{M} = \{1,2,...,m\}$, Initial vehicle resources $c_{max}^{\text{VES}}(t)$, Weight parameters $\gamma_1$, $\gamma_2$;
Ensure: Optimal matches of vehicles to grids.
1: Extract grids with mobile resource demands $w_i(t)$ as left nodes $i \in \mathscr{N}$, and vehicles with resource $e_j^{\text{VES}}(t)$ as right nodes $j \in \mathscr{M}$.
2: for each $ROUND$ do
3:   Initialize weight matrix $W \leftarrow \phi$
4:   for each left node $i \in \mathscr{N}$ do
5:     for each right node $j \in \mathscr{M}$ do
6:       Construct the distance matrix $d_{i,j}$
7:       if $d_{i,j} < D_{\max}$ then
8:         Perform BFS and calculate the shortest travel time $T_{i,j}$.
9:         if $T_{i,j} < T_{\max}$ then
10:          Form an edge with weight $W_{i,j} = \gamma_1 \times \min(e_j^{\text{VES}}, w_i(t)) - \gamma_2 \times d_{i,j}$.
11:        end if
12:      end if
13:    end for
14:  end for
15:  Perform KM matching.
16:  Update $e_j^{\text{VES}}(t)$ and $w_i(t)$.
17:  Re-extract grids with unmet demands as left nodes and unmatched vehicles as right nodes.
18: end for
19: return Optimal matches.
```

3) **Experimental Environment**: Our experiments were conducted on a workstation equipped with a 3.2 GHz Intel i9 CPU, 64 GB of memory, and an NVIDIA 3060 graphics card.

5.2 Forecasting Results

In the GICUNet model prediction, we set the hidden feature dimension of the network to 512 and use a batch size of 16. We choose the Lion optimizer for training with an initial learning rate of 0.002 and employ a cosine annealing strategy to dynamically adjust the learning rate. To train the GICUNet model, we utilize the mean squared error loss (MSELoss) to minimize the difference between the prediction and actual values.

We compare our prediction method with the current state-of-art methods TAU and SimVP on the TaxiBJ dataset, as well as the STResNet model designed specifically for the TaxiBJ dataset [30]. To ensure a fair comparison, we adjust the STResNet model by using data from the same preceding 7 days to match our experimental setup. Additionally, we use Avg as the reference group that obtains the averaged data from the preceding 7 days as the prediction results.

Table I shows the prediction results of the above methods. Our method's highest recorded computation cost is only 1.37G, significantly lower than TAU (6.12G), SimVP (9.42G), and STResNet (3.14G). Meanwhile, our method also maintains a high prediction accuracy, with an average MSELoss of 0.0155, significantly better than TAU (0.0185), SimVP (0.0190), and STResNet (0.0223).

Table 1. Comparison of Different Prediction Models.

	GICUNet	TAU	SimVP	STResNet
Computational Overhead (G)	1.37	6.12	9.42	3.14
Average MSELoss (M)	0.0155	0.0185	0.0190	0.0223

5.3 Computation Capacity of the Fixed Edge Severs

We set the unit deployment price of the computation capacity of the FES to p^{FES}, and the unit deployment price of the VES to p^{VES}, where p^{FES}:$p^{\mathrm{VES}} = 1$:2. Fig. 5 shows the computation capacity of the fixed edge severs and the minimum total cost through Bayesian optimization.

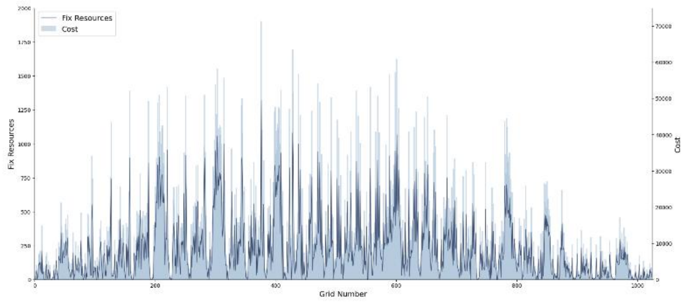

Fig. 5. Computation Capacity of the Fixed Edge Severs and Deployment Cost.

We compare the deployment cost of our BO-based method with two schemes:

1) *Half-Fixed*: We allocate half of the computing capacity to VESes and half of the computing capacity to FESes.

2) *All-Fixed*: We allocate all of the computing capacity to FESes, and it is entirely up to FES to provide computational offloading services to IoV users.

The results are shown in Table 2. Our approach yields much lower deployment costs than the comparison schemes: 7.5% lower compared with Half-Fixed and 31% compared with All-Fixed. This shows that our scheme can efficiently allocate computing resources and reduce the deployment cost.

Table 2. Deployment Cost of the Three Methods.

	BO	Half-Fixed	All-Fixed
Cost	1.33×10^7	1.43×10^7	1.72×10^7

5.4 Performance of Path Planning of the VESes

We use the demand obtained from the prediction results for path planning, and verify the performance of our deployment approach on the real demand. We will compare it with other three methods on the same dataset:

1) *MS-Single*: This algorithm performs one single iteration of the matching in the bipartite graph for path planning.
2) *High-Demand First*: This algorithm employs the greedy algorithm to send the VESes to high-demand grids in priority.
3) *Random Matching*: This algorithm matches VESes to grids randomly. All the VESes that meet the distance constraint have a chance to be matched.

We investigate two evaluating metrics: *service coverage* and *total travel distance*. Service coverage refers to the average ratio of the satisfied computation requests to the total number of requests of all the grids across 48 time slots per day. Total travel distance is defined as the sum of the travel paths of all the VESes across 48 time slots per day. It is worth noting that the total travel distance is calculated based on the paths we plan in advance, while the service coverage is computed to evaluate what we can achieve with the path planning scheme on real demand.

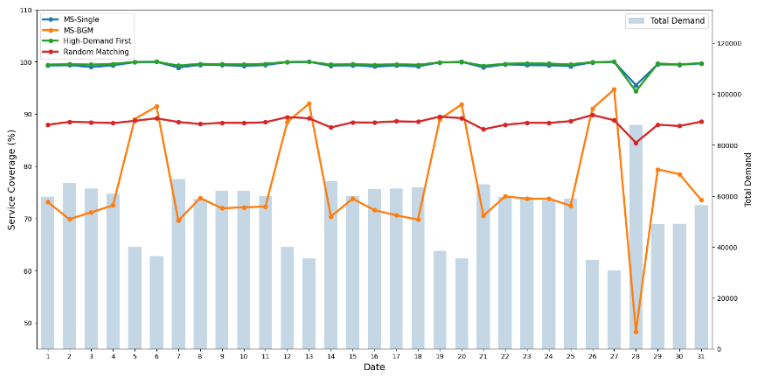

Fig. 6. Comparison of Service Coverage.

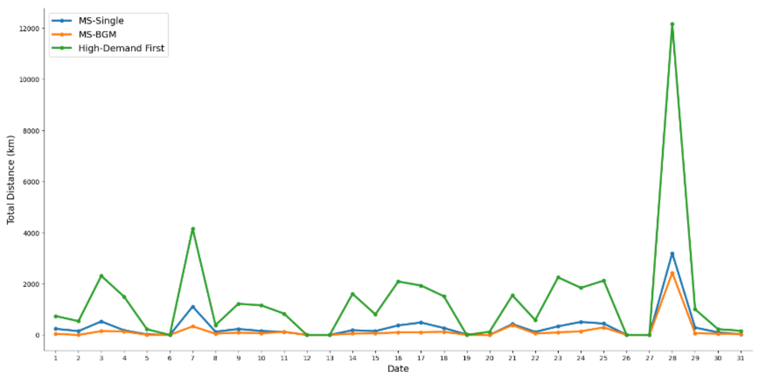

Fig. 7. Comparison of Total Travel Distance.

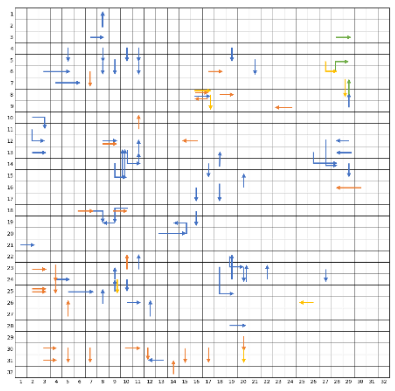

Fig. 8. An Example of the Real Vehicle Trajectories.

Fig. 6 presents the experimental results of the four methods on service coverage ratio. It can be observed that the demand is generally lower on weekends (e.g., March 5 and March 6), so the service coverage of almost all four methods can reach more than 90%. However, on weekdays with higher user demand, MS-Single, which only performs one single match between the VESes and grids, performs significantly worse than the other three methods, especially on March 28 when the user demand is exceptionally high. The service coverage of Random Matching is relatively stable, and it can reach about 80-90% regardless the user demand. Our MS-BGR algorithm and the High-Demand First method both demonstrate superior overall performance with stable and high service coverage, at least 10% higher compared with Random Matching and MS-Single.

Fig. 7 illustrates the comparison results of total travel distances among the four algorithms. The total travel distance for Random Matching is not shown in the figure as the it is too large compared with other methods. MS-Single has smaller total travel distance than other methods due to small number of matched VESes, but its service coverage is poor as discussed above. The High-Demand First method has a large total travel distance because it always greedily sends VESes to the higher-demand grids with higher demand, regardless of the travel distance. In contrast, our MS-BGR achieves the best performance among the four algorithms, e.g., 77% lower travel distance than High-Demand First.

Fig. 8 shows an example of the real trajectories of the VESes under our MS-BGR method on several time slots on a single day.

6 Conclusion and Future Work

In this study, we proposed a joint deployment scheme using Fixed and Vehicle-Mounted Edge Servers for Urban Internet of Vehicles. Firstly, leveraging historical traffic data, we designed the GICUNet traffic flow prediction model to accurately predict the future traffic in different areas. Subsequently, we applied Bayesian Optimization to determine the computation capacity of the FESes with the objective of minimizing the total deployment cost of the FESes and VESes. Then we designed a path planning algorithm for VESes to serve the grids and reduce energy consumption. The experimental results verified the superior performance of our proposed solution in terms of service coverage and total travel distance compared with other popular strategies.

In our work, the assumption of no inter-grid bottleneck may not fully capture real-world network dynamics, but our future extensions will explicitly model bandwidth allocation and congestion control. In addition, our VES energy replenishment mechanism and the homogeneous task assumption might be to idealistic. In our future research, we will extend the model to consider charging latency as well as heterogeneous task priorities, to further enhancing the robustness and adaptability of our solution.

Acknowledgments. This work was supported in part by the National Natural Science Foundation of China (Grant No. 62273107, No. 62371142), and GuangDong Basic and Applied Basic Research Foundation (Grant No. 2024A1515010404).

References

1. Raja, G., Manaswini, Y., Vivekanandan, G. D., Sampath, H., Dev, K., & Bashir, A. K. (2020, July). AI-powered blockchain-a decentralized secure multiparty computation protocol for IoV. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (pp. 865-870). IEEE.
2. Nguyen, T. D., Nguyen, V., Pham, V. N., Huynh, L. N., Hossain, M. D., & Huh, E. N. (2020). Modeling data redundancy and cost-aware task allocation in MEC-enabled Internet-of-Vehicles applications. *IEEE Internet of Things Journal*, *8*(3), 1687-1701.
3. Gui, G., Liu, M., Tang, F., Kato, N., & Adachi, F. (2020). 6G: Opening new horizons for integration of comfort, security, and intelligence. *IEEE Wireless Communications*, *27*(5), 126-132.
4. Li, S., Zhang, N., Lin, S., Kong, L., Katangur, A., Khan, M. K., ... & Zhu, G. (2018). Joint admission control and resource allocation in edge computing for internet of things. *IEEE Network*, *32*(1), 72-79.

5. Anwar, M. R., Wang, S., Akram, M. F., Raza, S., & Mahmood, S. (2021). 5G-enabled MEC: A distributed traffic steering for seamless service migration of internet of vehicles. *IEEE Internet of Things Journal*, *9*(1), 648-661.
6. Shi, W., Sun, H., Cao, J., Zhang, Q., & Liu, W. (2017). Edge computing-an emerging computing model for the internet of everything era. *Journal of computer research and development*, *54*(5), 907-924.
7. Bozorgchenani, A., Tarchi, D., & Corazza, G. E. (2018). Centralized and distributed architectures for energy and delay efficient fog network-based edge computing services. *IEEE Transactions on Green Communications and Networking*, *3*(1), 250-263.
8. Chen, Y., Zhao, F., Chen, X., & Wu, Y. (2021). Efficient multi-vehicle task offloading for mobile edge computing in 6g networks. *IEEE Transactions on Vehicular Technology*, *71*(5), 4584-4595.
9. Wang, J., Liu, K., & Pan, J. (2019). Online UAV-mounted edge server dispatching for mobile-to-mobile edge computing. *IEEE Internet of Things Journal*, *7*(2), 1375-1386.
10. Zhan, C., Hu, H., Liu, Z., Wang, Z., & Mao, S. (2021). Multi-UAV-enabled mobile-edge computing for time-constrained IoT applications. *IEEE Internet of Things journal*, *8*(20), 15553-15567.
11. Yi, C., Cai, J., Zhang, T., Zhu, K., Chen, B., & Wu, Q. (2021). Workload re-allocation for edge computing with server collaboration: A cooperative queueing game approach. *IEEE Transactions on Mobile Computing*, *22*(5), 3095-3111.
12. Bo, J., & Zhao, X. (2025). Vehicle Edge Computing Task Offloading Strategy Based on Multi-Agent Deep Reinforcement Learning. *Journal of Grid Computing*, *23*(2), 1-32.
13. Gong, C., Wei, L., Gong, D., Li, T., & Feng, F. (2022). Energy-Efficient Task Migration and Path Planning in UAV-Enabled Mobile Edge Computing System. *Complexity*, *2022*(1), 4269102.
14. Du, X., Guo, Q., Zhang, Y., Li, Y., & Zhou, Y. (2022). Multi-UAV Cooperative Assisted RSU Data Acquisition Strategy considering Coverage Quality. *Wireless Communications and Mobile Computing*, *2022*(1), 5383526.
15. Al-Hilo, A., Samir, M., Assi, C., Sharafeddine, S., & Ebrahimi, D. (2021). A cooperative approach for content caching and delivery in UAV-assisted vehicular networks. *Vehicular Communications*, *32*, 100391.
16. Saurez, E., Gupta, H., Daglis, A., & Ramachandran, U. (2021, November). Oneedge: An efficient control plane for geo-distributed infrastructures. In *Proceedings of the ACM Symposium on Cloud Computing* (pp. 182-196).
17. AlQerm, I., & Pan, J. (2021). DeepEdge: A new QoE-based resource allocation framework using deep reinforcement learning for future heterogeneous edge-IoT applications. *IEEE Transactions on Network and Service Management*, *18*(4), 3942-3954.
18. Chang, L., Deng, X., Pan, J., & Zhang, Y. (2021). Edge server placement for vehicular ad hoc networks in metropolitans. *IEEE Internet of Things Journal*, *9*(2), 1575-1590.
19. Li, X., Huang, A., Fan, Y., Tang, Z., Chang, L., & Wang, T. (2024, October). Elastic and Energy-Efficient Virtual Machine Allocation on Edge Servers for Internet of Vehicles. In *2024 IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA)* (pp. 1058-1064). IEEE.
20. Yang, L., Yao, H., Wang, J., Jiang, C., Benslimane, A., & Liu, Y. (2020). Multi-UAV-enabled load-balance mobile-edge computing for IoT networks. *IEEE Internet of Things Journal*, *7*(8), 6898-6908.
21. Mishra, D., & Natalizio, E. (2020). A survey on cellular-connected UAVs: Design challenges, enabling 5G/B5G innovations, and experimental advancements. *Computer Networks*, *182*, 107451.

22. Liu, B., Zhang, W., Chen, W., Huang, H., & Guo, S. (2020). Online computation offloading and traffic routing for UAV swarms in edge-cloud computing. *IEEE Transactions on Vehicular Technology*, *69*(8), 8777-8791.
23. Wang, Y., Tang, Z., Huang, A., Zhang, H., Chang, L., & Pan, J. (2024). Placement of UAV-mounted edge servers for internet of vehicles. *IEEE Transactions on Vehicular Technology*.
24. Feng, J., Liu, Z., Wu, C., & Ji, Y. (2018). Mobile edge computing for the internet of vehicles: Offloading framework and job scheduling. *IEEE vehicular technology magazine*, *14*(1), 28-36.
25. Liu, Y., Yu, H., Xie, S., & Zhang, Y. (2019). Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks. *IEEE Transactions on Vehicular Technology*, *68*(11), 11158-11168.
26. Liu, L., Feng, J., Mu, X., Pei, Q., Lan, D., & Xiao, M. (2023). Asynchronous deep reinforcement learning for collaborative task computing and on-demand resource allocation in vehicular edge computing. *IEEE Transactions on Intelligent Transportation Systems*, *24*(12), 15513-15526.
27. Tang, Z., Huang, A., Wang, Y., Chang, L., & Wang, T. (2023, December). Edge Servers on Wheels: Deployment and Route Planning of Mobile Servers for Internet of Vehicles. In *2023 19th International Conference on Mobility, Sensing and Networking (MSN)* (pp. 707-713). IEEE.
28. Deng, X., Bao, Y., Chen, S., & Chang, L. (2024, November). Computation Offloading with Bus-Mounted Edge Servers Based on Dung Beetle Optimization in Internet of Vehicles. In *2024 Twelfth International Conference on Advanced Cloud and Big Data (CBD)* (pp. 60-65). IEEE.
29. Kuang, L., Hua, C., Wu, J., Yin, Y., & Gao, H. (2020). Traffic volume prediction based on multi-sources GPS trajectory data by temporal convolutional network. Mobile Networks and Applications, 25, 1405-1417.
30. Zhang, J., Zheng, Y., & Qi, D. (2017, February). Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 31, No. 1).