Spotlighter: Backup Age-Guaranteed Immersive Virtual Vehicle Service Provisioning in Edge-Enabled Vehicular Metaverse

Yu Qiu, Min Chen, *Fellow, IEEE*, Hebin Huang, Weifa Liang, *Senior Member, IEEE*, Junbin Liang, Yixue Hao, *Member, IEEE*, and Dusit Niyato, *Fellow, IEEE*

***Abstract*—Edge-enabled Vehicular Metaverse (EVM) is a new paradise supported by various compute-intensive Virtual Vehicle Services (VVSs), where users can immerse and enjoy their spiritual world. User immersion is critical during VVS provisioning in the EVM, yet it can be weakened or curtailed by a sense of disengagement caused by unknown failures. Providing redundant backups VVSs (BVVSs) and keeping the Age of Backup Information (AoBI) could effectively resist and avoid this disengagement when failures occur. However, the trajectories of mobile vehicles are unknown and dynamic, which makes it challenging to optimally migrate VVSs and BVVSs or adjust the update frequency of backup information in real-time, so as to ensure service reliability and AoBI while minimizing the cost of accepting VVS-based metaverse services. In this paper, the above long-term issue is first decomposed into discrete single-slot sub-problems that are modeled as integer linear programming problems. Then, a comprehensive resource explorer named spotlighter is designed, where the first and second parts are a metaverse service home prediction algorithm based on deep learning and a VVS migration algorithm based on randomized rounding, respectively. By tracking the dynamical locations of service homes based on current and historical information, the former can help the latter to adaptively minimize migration costs on VVS re-instantiation and traffic transmission among services and moving vehicles. Finally, a cost-adaptive AoBI guarantee algorithm is merged in spotlighter to ensure the freshness of backup status, by trading-off synchronization cost on BVVS migration, backup update, and backup synchronization. Theoretical analyses and experiments based on real databases show that our algorithms are promising compared with baseline algorithms.**

***Index Terms*—Age of information, Internet of Vehicles, metaverse, mobile edge cloud, network function virtualization, reliability.**

Manuscript received 14 July 2023; revised 24 April 2024; accepted 7 July 2024. Date of publication 10 July 2024; date of current version 5 November 2024. The work of Min Chen was supported in part by the National Natural Science Foundation of China under Grant 62276109 and Grant 62176101 and in part by Guangdong Basic and Applied Basic Research Foundation under Grant 2024A1515030017 and Grant 2024A1515011153. The work of Weifa Liang was supported by Hong Kong Research Grants Council (RGC) under Grant 7005845, Grant 8730094, Grant 9043510, and Grant 9380137. The work of Dusit Niyato was supported in part by the National Research Foundation, Singapore, and Infocomm Media Development Authority under its Future Communications Research & Development Programme, Defence Science Organisation (DSO) National Laboratories under the AI Singapore Programme under AISG Award AISG2-RP-2020-019 and Award FCP -ASTAR-TG-2022-003, and in part by Singapore Ministry of Education (MOE) Tier 1 under Grant RG87/22. Recommended for acceptance by H. Shen. *(Corresponding authors: Min Chen; Yixue Hao.)*

Yu Qiu and Min Chen are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510640, China, and also with Pazhou Laboratory, Guangzhou 510640, China (e-mail: csqiuyu@mail.scut.edu.cn; minchen@ieee.org).

Hebin Huang and Junbin Liang are with the Guangxi Key Laboratory of Multimedia Communications and Network Technology, School of Computer, Electronics and Information, Guangxi University, Nanning 118, China.

Weifa Liang is with the Department of Computer Science, City University of Hong Kong, Hong Kong, China.

Yixue Hao is with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 1037, China (e-mail: yixuehao@hust.edu.cn).

Dusit Niyato is with the College of Computer Science and Engineering, Nanyang Technological University, Singapore 639798.

This article has supplementary downloadable material available at https://doi.org/10.1109/TMC.2024.3425896, provided by the authors.

Digital Object Identifier 10.1109/TMC.2024.3425896

I. Introduction

A. Background

THE vehicular metaverse for users in mobile vehicles is an immersive and ubiquitous virtual cyberspace, converging the physical and multiple digital worlds by various Virtual Vehicle Services (VVSs) [1], such as autonomous driving, companion, and tourism services, etc. To ensure cloud-like service quality and reduce cost and latency during metaverse service provisioning enabled by VVSs, edge intelligence, as one of the most fundamental and complementary technologies, is converged into the vehicular metaverse. It can push computing, storage, and network resources to network edge in proximity to metaverse users [2]. The edge-enabled vehicular metaverse (EVM), with a unique human-centric feature, is revolutionizing patterns of VVS provisioning in many areas ranging from entertainment to industry [3]. Humans in mobile vehicles are the main players who operate digital proxies called avatars to request various VVS-enabled metaverse tasks for achieving immersive second life in corresponding digital worlds. For instance, NISSAN company plans to fully implement the “invisible-to-visible” system[1] by 2025 for vehicles. The system uses a 3D augmented reality interface to make information invisible to drivers visible for improving driving comfort and convenience.

To release the potential and benefits of the human-centric EVM, user immersion, which is a challenging issue overlooked by traditional networks, needs to be solved with high priority. User immersion in EVM refers to the experience of users being transported to elaborately simulated virtual worlds, where our

[1]https://www.nissan-global.com/JP/INNOVATION/TECHNOLOGY/ARCHIVE/I2V/

brains are surrounded by VVS-enabled metaverse tasks with an intensity that can blur the world around us. In other words, user immersion is divided into three levels from bottom to top: 1) participation in metaverse tasks based on various VVSs; 2) lack of time sensitivity; and 3) loss of space (real world) awareness. For instance, Bob, a passenger in a car, initiates the companion task during automatic driving, and participates in basic conversation. As the discussion deepens, Bob is increasingly interested in the content, losing track of time. Meanwhile, Bob also utilizes the mixed reality assistance to invoke multiple senses, such as sight, touch, and so on. As he engages more deeply with the interaction experience, his attention will focus on the virtual conversation, and lose the awareness of his surroundings including passing scenery. However, unknown failures are the main barrier to user immersion, since any VVS failure due to hardware or software faults could invalidate the whole virtual world, and suddenly pull users away from the metaverse task. This not only immediately interrupts the participation process of users during VVSs-based metaverse tasks, but even causes mental uncomfortableness to users due to the difference and suddenness of the sense of time and space. Even worse, some failures may occur in VVS-related driving decisions, which can lead to traffic accidents.

B. Motivations

An immersive VVS-enabled metaverse service provisioning scheme in the EVM needs to consider both failure prevention and backup status synchronization mechanisms for resisting unknown failures. In terms of prevention mechanisms, the placement of additional backup VVSs (BVVs) near VVSs in advance could improve the reliability of VVSs-based metaverse tasks. When a VVS suddenly fails due to software or hardware, users in mobile vehicles can continue enjoying the metaverse service simply by activating these pre-placed idle backups. Nevertheless, in a dynamic EVM network, the above mechanism may be ineffective and contradicted when service failures occur, because these backups only have outdated status information about vehicles and environments from their primary VVSs.

The backup status synchronization mechanism is considered as a complement to the prevention mechanism, which adaptively adjusts the update rates of backups based on specific service scenarios and the distances between the pair of primary-backup services to ensure the Age of Backup Information (AoBI), where the AoBI is defined as the amount of time elapsed since a backup VVS received the last backup status information from the corresponding primary VVS. This ensures the reliability of improved VVS-based metaverse services and avoids the impact of stale backup information during high-speed driving. Unlike other age-related concepts including age of information (AoI) or age of data package [4], AoBI focuses on a one-to-many and location-variable environment, i.e. 1) there are one source (VVS) and multiple destinations (BVVSs), and 2) both the locations of the source and destinations are changeable overtime.

However, the trajectories of mobile vehicles are unknown and dynamic in a resource-constrained distributed EVM. In addition, the demands of users on metaverse service reliability, accuracy, and type are always distinguishing during driving. Even worse, ill-conceived migration, activation, and synchronization of VVSs and BVVSs would exhaust the limited resources of edge servers, which would further affect other user immersions due to the inability to offload VVS-enabled metaverse services to edge servers. Given these limitations in the resources-constrained EVM, how to implement immersive VVSs-enabled metaverse service provisioning poses serious challenges: 1) how to capture the dynamic location of service homes, namely the nearest access points, for every mobile vehicle in real-time without any prior information, such as moving trajectory, 2) how to optimally migrate VVSs to selected edge services to follow moving vehicles, 3) how to calculate BVVS numbers required for each VVS and where they are deployed, 4) how to optimally migrate backup or adjust update frequency of backup information to ensure AoBI during driving.

C. Contributions

The novelty of this paper lies in the first pilot exploration of immersive VVS provisioning in an EVM environment where each VVS data is equally important yet backup data is significant only if it is freshest, and each vehicle with given demands on service type, accuracy, reliability, and backup freshness. A practical yet reliable immersion service provisioning explorer, namely spotlighter, is designed to make each metaverse service always in the best state. The spotlighter can first predict the suitable services home for all moving vehicles in real-time, and then ensure service reliability and the AoBI of corresponding backups.

The main contributions of this paper are as follows:

- To the best of our knowledge, we are the first to formulate an integer linear programming for the immersive VVS-enabled metaverse service provisioning problem from a new insight that jointly considers vehicle mobility, VVS reliability, and the age of backup information. We also show an offline version of the problem is NP-hard.
- To provide a feasible solution to this NP-hard problem, a learning-driven metaverse service home prediction algorithm for sensing which access point will be connected to cars in the next slot is developed. Meanwhile, by using randomization and probabilistic techniques, a VVS migration algorithm with a provable approximation ratio is devised to minimize the total VVS migration cost, and its solution enables providing continuous immersion in VVS-enabled metaverse services.
- To ensure the reliability and AoBI for all services before failures occurred, we design a heuristic algorithm with worst-case performance bound to adaptively select the backup migration strategy or the backup update strategy by trading off the BVVS migration cost, backup updating cost, and the backup synchronization cost.
- To evaluate the effectiveness of the proposed algorithms, we conduct extensive simulation experiments based on a real vehicle dataset. The simulation results show that our algorithms can reduce the total service cost which consists of the migration cost and the synchronization cost, by at least 10.28% and effectively adapt to the EVM network.

The rest of this article is organized as follows. Section II introduces the EVM network, reliability, AoBI, receiving cost, and other mathematical models. Section III introduces the definition, NP-hardness, and formulation of the research problem. Section IV introduces the metaverse service home prediction algorithm. In Section V, the VVS migration algorithm and mathematical analyses are introduced. Section VI shows the related algorithm of the cost-adaptive AoBI guarantee algorithm. Section VII evaluates the performances of our algorithms. Section VIII reviews related works. Section IX summarizes the full paper.

II. Preliminaries

In this section, we introduce the system model, notions, and notations.

A. EVM Network and Vehicle Mobility Model

An edge-enabled vehicular metaverse (EVM) network can be modeled by an undirected weighted graph $G = (V, E)$, where the set V of vertices can be further divided into two disjoint subsets: a set of access points V_a co-located with edge servers and a set V_c of mobile vehicles. Each edge server $v_e \in V_e$ has computing capacity Cap_{ve}, but an access point and a mobile car $v_c \in V_c$ are responsible for data transfer only. Note that both edge servers and access points are stationary, while each vehicle is movable in its own driving direction at a dynamic speed $S_i(t)$ meters per second.

Denote by $L_i(t)$ and $L^j_{V_a}$ the location of vehicle car_i at time slot t and the location of access point AP_j, respectively. The edge set E is the union of a set E_s of static links among access points, and the set of dynamic links between vehicles and access points. Note that there is a dynamic link between a vehicle and an access point if the vehicle v_c is within the transmission range D_{V_a} of the access point, namely $\|L_i(t) - L^j_{V_a}\| \leq D_{V_a}$. In addition, when a mobile vehicle is under the convergence of multiple access points, the vehicle will connect to the closest access point.

B. Virtual Vehicle Service-Enabled Metaverse Request Model

Let $F = \{f_i \mid 1 \leq i \leq |F|\}$ be the set of all different types of Virtual Vehicle Services (VVS) in the EVM network, where each service can be supported by a series of AI models with similar architectures but different parameters, such as VGG-11, VGG-13, VGG-16, and VGG-19. For a service based on the same series of AI models, the accuracy of the service increases with the number of parameters in a selected model [5]. The service $f_i \in F$ demands the amount $Cap^{max}_{f_i}$ of computing resource when adopting the model with the most accuracy. Following the experimental results in [6] that the amount of computing resource demanded by a service based on the same series of models (e.g., VGG, ResNet, and DualPathNet) is linearly correlated to the accuracy of that model provided, the amount Cap_{f_i} of computing resource of a VVS-enabled metaverse service with accuracy $\lambda_i \leq 1$ thus is as follows [7]:

$$Cap_{f_i} = \lambda_i \cdot Cap^{max}_{f_i}. \tag{1}$$

TABLE I
Notations

Notation	Definition
G	The edge-enabled vehicular metaverse network
V_e and V_c	The set of edge servers and car in G, respectively
E	The set of links between vertices in G
Cap_{ve}	The computing capacity of edge server v_e
Cap_{f_i}	The demanding computing resource for a VVS f_i
γ_i	The VVS-enabled metaverse request issued by car_i
$AP_i(t)$	The server connected of the vehicle at time slot t
R_i	The user requirement for VVS reliability in γ_i
λ_i	The user requirement for VVS accuracy in γ_i
θ	The user requirement for AoBI in γ_i
r	The initial reliability of a VVS
$\hat{r}$	The enhanced reliability of a VVS through its backups
$T^S_{i,j}$	The generation time for the jth update packet in γ_i
$T^T_{i,j}$	The transmission time for the jth update packet in γ_i
$T^D_{i,j}$	The received time for the jth update packet in γ_i
$\mathfrak{f}^i_b$	The update frequency of backup information for γ_i
$\mathfrak{f}^{i,j}_b$	The expected update frequency of jth backup in γ_i
$C^I_{\gamma_i}$	The VVS re-instantiation cost for γ_i
$C^T_{\gamma_i}$	The VVS traffic migration cost for γ_i
$C^{IB}_{\gamma_i}$	The BVVS migration cost for γ_i
$C^U_{\gamma_i}$	The backup updating cost for γ_i
$C^S_{\gamma_i}$	The backup synchronization cost for γ_i
$x_{i,v,u}$	Whether VVS requested by γ_i is migrated from V_{e_v} to V_{e_u}
$y^i_{b_j,v,u}$	Whether jth backup in γ_i re-instantiate at V_{e_u} for γ_i
$y^i_{b_j,v,vb_j}$	Whether jth backup in γ_i synchronizate from V_{e_v} to $V_{e_{vb_j}}$
$y^{i,j}_{u,v}$	Whether an update packet of jth backup in γ_i uses link $e_{u,v}$
z^i_b	Whether an age violation has occurred in γ_i

When the service accuracy, reliability, and AoBI of a user can be met simultaneously, the user is regarded as immersed in the virtual metaverse world. A VVS-enabled metaverse request γ_i issued by a mobile vehicle car_i can be represented by a tuple

$$\gamma_i = (L_i(t), S_i(t), AP_i(t), F_i, R_i, \lambda_i, \theta), \tag{2}$$

where $L_i(t)$ and $S_i(t)$ represent the location and the driving speed of the vehicle at time slot t, and $AP_i(t)$ is the server connected at time slot t. F_i represents the index of a single VVS f_{F_i} requested by a mobile vehicle in F, where f_{F_i} is simply referred to as f_i in the following. R_i is the reliability requirement for the VVS f_i, λ_i and θ represent accuracy and AoBI requirements. Denote by $\mathcal{R} = \bigcup_{i=1}^{|V_c|}\{\gamma_i\}$ the set of all VVS requests by mobile vehicles in the EVM network. For the sake of convenience, in the rest of this paper, we use index i to refer to all variables related to mobile vehicle car_i.

C. Reliability Model

The running state of a VVS could be in a normal or faulty state due to unpredictable failures of software and/or hardware. The reliability r of a VVS is defined as the ratio of the VVS uptime, namely the time in the normal state, to the total running time,

$$r = \frac{T_{uptime}}{T_{all}} = \frac{MTTN}{MTTN + MTTF}, \tag{3}$$

where $MTTN$ and $MTTF$ are the mean duration to normal and mean time to the fault duration of the service, and the fault duration can be obtained based on historical system data and logs [8]. In addition, VVS fault is an antithetical event to VVS normal service, whose definition is given as follows:

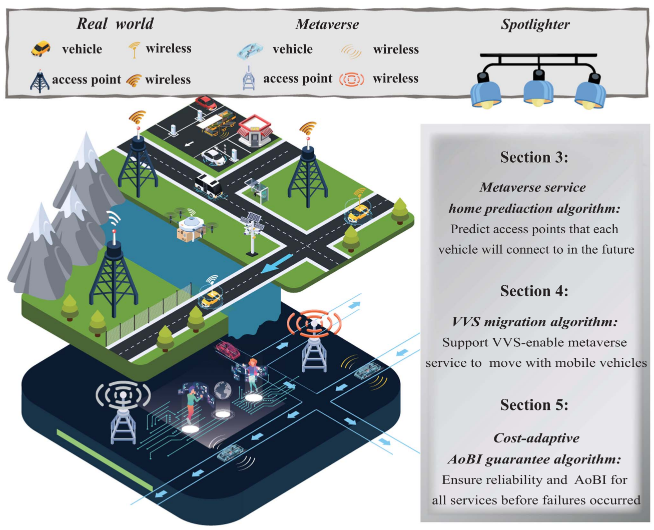
$$\rho = 1 - r, \tag{4}$$

where r is the reliability and ρ is the failure rate of the VVS. To ensure continuity and reliability of VVS-enabled metaverse service, additional backups need to be added before a fault occurs, and activated to actively take over traffic when one occurs. The VVS requested by a mobile vehicle is failure only if the primary and its all backup VVSs fail at the same time. We assume that each VVS failure occurs independently and does not affect each other, so the fault state of each VVS is independent and follows the i.i.d. over time. Therefore, the enhanced reliability $\hat{r}$ of a VVS through its backups, and the minimum number k of backups needed is calculated as follows:

$$\hat{r} = 1 - \rho_p \cdot \prod_{b=1}^{k} \rho_b, \tag{5}$$

$$R_i \leq \hat{r} \;\Rightarrow\; k \geq \left\lceil \log_{\rho_b} \frac{1 - R_i}{\rho_p} \right\rceil, \tag{6}$$

where ρ_p and ρ_b are the failure rates of VVS and BVVS, respectively.



Real world
vehicle
wireless
access point
wireless
Metaverse
vehicle
wireless
access point
wireless
Spotlighter
Section 3:
Metaverse service home prediaction algorithm:
Predict access points that each vehicle will connect to in the future
Section 4:
VVS migration algorithm:
Support VVS-enable metaverse service to move with mobile vehicles
Section 5:
Cost-adaptive AoBI guarantee algorithm:
Ensure reliability and AoBI for all services before failures occurred

Fig. 1. An EVM network.

D. Age of Backup Information Model

State-sensitive control metaverse services are required by vehicles during driving, such as collision warnings based on cooperative awareness messages and VR/XR. To ensure that users are safely immersed in the EVM, the reliability and the Age of Backup Information (AoBI) of these security functions should be also the top priorities. All state data of a VVS is first packed into one update packet, and then the packet is sent to BVVSs that need to be updated.

Let $T_{i,j}^S$ be the generation time of the jth update packet by its primary service in VVS request γ_i issued by car_i, and let $T_{i,j}^D$ be the received time of the packet by one of its backups. In the interval between $T_{i,j}^S$ and $T_{i,j}^D$, the update packet transmission from the primary VVS to a BVVS incurs the transmission time $T_{i,j}^T$,

$$T_{i,j}^T = \frac{1}{3 \times 10^5} \sum_{u,v \in V} e_{u,v} \cdot y_{u,v}^{i,j}(t), \tag{7}$$

where $y_{u,v}^{i,j}(t)$ is a binary variable, indicating whether the update packet makes use of link $e_{u,v}$ between edge servers V_{e_u} and V_{e_v} at time slot t. Thus, $T_{i,j}^D = T_{i,j}^S + T_{i,j}^T$. After a BVVS receives the update packet at time $T_{i,j}^D$, the VVS undergoes a waiting time $T_{i,j}^W$ before sending the next update at slot $T_{i,j+1}^S$ [9], namely $T_{i,j+1}^S = T_{i,j}^D + T_{i,j}^W$. Note that the data sensed by vehicle sensors is collected and processed by the VVS continuously, yet the backup information of the VVS is generated before the AoBI of the BVVS is about to violate. So, the inter-delivery time $T_{i,j}^Y$ between two consecutive update packets is controlled by the update frequency $\mathfrak{f}_b^i$ [10] of VVS f_i, and is denoted as follows:

$$T_{i,j}^Y = T_{i,j+1}^S - T_{i,j}^S = \frac{1}{\mathfrak{f}_b^i(t)}, \tag{8}$$

and

$$T_{i,j}^W = T_{i,j}^Y - T_{i,j}^T, \tag{9}$$

where the transmission time from the primary VVS to its backup and the backup frequency $\mathfrak{f}_b^i(t)$ would fluctuate over time due to the random movement of vehicles. In order to avoid network congestion, all VVSs are limited by the maximum update frequency $\mathfrak{f}_{max}$. Compared to $T_{i,j}^T$ and $T_{i,j}^W$, the sampling of backup information can be done in a relatively short time and is usually negligible [11]. In this context, sampling and updating are the same, and they are used interchangeably.

The AoBI $\Delta_b^i(t)$ represents the time elapsed since the information generation time of the latest backup status update at the backup service, defined as follows:

$$\Delta_b^i(t) = t - \max\{T_{i,j}^S \mid T_{i,j}^D \leq t\}, \tag{10}$$

where $\max\{T_{i,j}^S \mid T_{i,j}^D \leq t\}$ is the time slot of the most fresh (jth) update packet generated. Because $T_{i,j}^D \leq T_{i,j+1}^D$, $\Delta_b^i(t)$ can be also written as:

$$\Delta_b^i(t) = t - T_{i,j}^S, \text{ if } T_{i,j}^D \leq t < T_{i,j+1}^D. \tag{11}$$

The specific evolution process of AoBI for a single backup is shown in Fig. 2, whose mathematical model is below.

$$\Delta_b^i(t+1) = \begin{cases} \Delta_b^i(t) + 1, & a_b^i(t+1) = 0, \\ T_{i,j}^T, & a_b^i(t+1) = 1. \end{cases} \tag{12}$$

$$a_b^i(t) = \mathbb{1}\left[max(T_{i,j}^D) \leq t\right]. \tag{13}$$

Specifically, the AoBI linearly increases over time until the jth update packet is received by the backup service, at which its AoBI is updated to the value of transmission time of the packet, as shown by the blue line in Fig. 2. On the basis of the evolution process, the AoBI can be updated iteratively over time according to the following formula,

$$\Delta_b^i(t+1) = a_b^i(t) \cdot T_{i,j}^T + (1 - a_b^i(t))(\Delta_b^i(t) + 1). \tag{14}$$

In the multi-backup EVM scenario, each primary VVS-enabled metaverse service has one and more backups. The AoBI of the primary service is determined by the largest value among

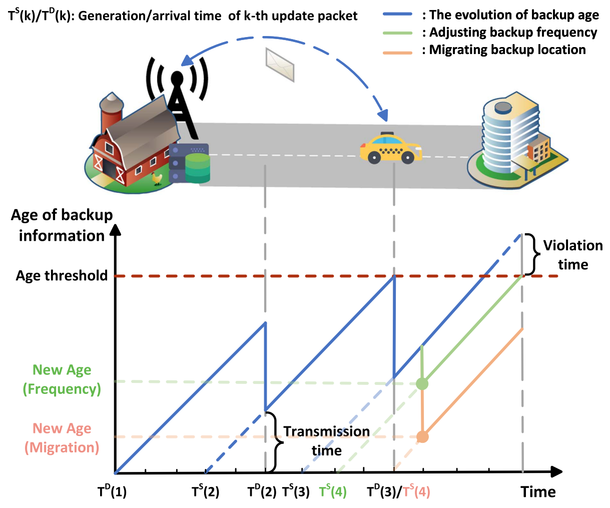


Fig. 2. The evolution of AoBI in a single backup scenario.

corresponding backups.

$$\Delta^i_{max}(t) = \max_j \Delta^{i,j}_b(t). \tag{15}$$

If a vehicle updates its backup status at a fixed frequency during driving, its backup age may increase inevitably and even exceed a user's tolerance threshold θ. Define $z^i_b(t)$ as a binary variable that indicates whether an age violation has occurred.

$$z^i_b(t) = \begin{cases} 1, & \theta \leq \Delta^i_{max}(t), \\ 0, & \theta > \Delta^i_{max}(t). \end{cases} \tag{16}$$

The freshness of the backup status can be reduced by adjusting the frequency of backup updating and migrating the backup near the primary services. As Fig. 2 shows, the green lines reduce backup age by increasing the update frequency in $T^S(3)$ slots (one update from 3 slots to one update in single slot). The frequency update formulas (17) and (18) are as follows (see Theorem 3 for details), where $\mathfrak{f}^{i,j}_b(t)$ is the expected frequency of jth backup, and $\mathfrak{f}^i_b(t)$ is the actual update frequency of VVS at t slot.

$$\frac{1}{T^T_{i,j}(t)} \geq \mathfrak{f}^{i,j}_b(t) \geq \frac{1}{\theta - T^T_{i,j}(t)}. \tag{17}$$

$$\mathfrak{f}^i_b(t) = \max_j \mathfrak{f}^{i,j}_b(t). \tag{18}$$

In addition, the pink line can also achieve the same goal by migrating backups (reduced transmission times to 1 slot) in $T^D(3)$ slots.

E. Cost Model

In the EVM, dynamic movements of vehicles cause real-time vehicle location to change over time. When a vehicle car_i moves out of the coverage area D_{V_a} of the current access point AP_v and into that of the next access point AP_u ($\|L_i(t) - L^v_{V_a}\| > D_{V_a}$ and $\|L_i(t) - L^u_{V_a}\| \leq D_{V_a}$), or AP_v is not the closest access point to car_i, VVS-enabled metaverse service and its backup need to follow moving vehicles, and the backup information is updated and synchronized among them in a timely manner to ensure the seamless and reliability of services. Therefore, five types of costs are considered in this section: VVS re-instantiation cost, traffic migration cost, BVVS migration cost, backup updating cost, and backup synchronization cost.

Migration cost: During the migration of metaverse services running on the server co-located with AP_v to the server co-located with AP_u, the cost of migration that consists of VVSs re-instantiation cost and traffic migration cost is considered at this slot t.

(1) VVSs re-instantiation cost: When vehicles run into the communication coverage of a server without running the VVS that they need, such as AP_u, or AP_v is not the closest access point to vehicles, these services have to be redeployed on this server co-located with AP_u. Define c_c as the unit cost of computing resources. Therefore, the re-instantiation cost C^I_i of VVS f_i with Cap_{f_i} computing demand is as follows:

$$C^I_{\gamma_i}(t) = C^I_i \cdot x_{i,v,u} = c_c \cdot Cap_{f_i} \cdot x_{i,v,u}, \tag{19}$$

where $x_{i,v,u}$ is a binary variable that indicates whether the VVS requested by γ_i is migrated from server V_{e_v} to server V_{e_u}.

(2) Traffic migration cost: Typically, the cost of the traffic transmission is proportional to the length of a link [12]. Let $P_{v,u}$ be the shortest path in G between AP_v and AP_u that is used for service migration, and let $P_{u,v'}$ be the shortest path in G between AP_u and $AP_{v'}$ that is the nearest a car for data transmission. The concatenation of these two paths forms a new path $P_{v,u,v'}$ with at least 2 hops between v and v'. Therefore, the traffic migration cost of a single vehicle $C^T_{\gamma_i}(t)$ at time slot t is as follows:

$$C^T_{\gamma_i}(t) = \frac{c_e B_i}{2} \cdot (|P_{v,u}| + |P_{u,v'}|) \cdot x_{i,v,u}, \tag{20}$$

where B_i is the packet rate of each vehicle car_i, and c_e is the unit prices for links.

Synchronization cost: During the migration of metaverse services along with mobile vehicles, the AoBI may violate. In addition to reducing transmission latency by moving the BVVS closer to vehicles to avoid violations, the same can be achieved by increasing the frequency of backup updates. The cost of synchronization that consists of BVVS migration cost, backup updating, and backup synchronization cost is considered at this slot t.

(1) BVVS migration cost: Because the essence of BVVS is idle VVS, it is activated when the VVS fails, the computing demand of BVVSs is the same as that of the corresponding VVS. Therefore, BVVS migration cost C^I_{ib} is only equal to VVS re-instantiation cost, regardless of the traffic costs, defined as follows:

$$C^I_{ib} = c_c \cdot Cap_{f_i}. \tag{21}$$

To reduce the cost of backup re-instantiations, only selected individual backups are migrated for each vehicle. Set a binary variable $y^i_{b_j,v,u}$ to indicate whether the jth backup of vehicle car_i is selected for re-instantiation at server V_{e_u}. Therefore, the total cost $C^{IB}_{\gamma_i}(t)$ of backup re-instantiation of car_i at t slot is as follows:

$$C^{IB}_{\gamma_i}(t) = \sum_{1\le j\le k} C^I_{ib} \cdot y^i_{b_j,v,u}. \tag{22}$$

(2) Backup updating cost: The size of backup status information generated by a VVS is proportional to its computing resource demand Cap_{f_i}, and the proportional coefficient is α [13], [14]. Set a binary variable $y^i_{b_j,v,vb_j}$ to indicate whether the jth backup of vehicle car_i is selected for synchronization from server V_{e_v} to server $V_{e_{vb_j}}$. Therefore, the backup updating cost $C^U_{\gamma_i}(t)$ of a single vehicle at time slot t is as follows:

$$C^U_{ib_j} = c_c \cdot \alpha \cdot Cap_{f_i}, \tag{23}$$

$$C^U_{\gamma_i}(t) = \sum_{1\le j\le k} C^U_{ib_j} \cdot y^i_{b_j,v,vb_j}. \tag{24}$$

(3) Backup synchronization cost: Let P_{v,vb_j} be the shortest path from AP_v that runs a VVS to AP_{vb_j} that runs corresponding jth backup. The backup synchronization cost $C^S_{ib_j}(t)$ of this backup is

$$C^S_{ib_j}(t) = \sum_{e\in P_{v,vb_j}} c_e \cdot e \cdot \alpha \cdot Cap_{f_i}. \tag{25}$$

To avoid transmitting massive redundant information about backup status, only selected individual backups are synchronized for each vehicle. Thus, the backup synchronization cost $C^S_{\gamma_i}(t)$ of a single vehicle is

$$C^S_{\gamma_i}(t) = \sum_{1\le j\le k} C^S_{ib_j}(t) \cdot y^i_{b_j,v,vb_j}. \tag{26}$$

All in all, the total acceptance cost of $C_{\gamma_i}(t)$ the service request of vehicle car_i at t slot is the sum of migration and synchronization costs, where $\mathfrak{f}^i_b(t)$ is the backup updating frequency. C_{all} is the total acceptance cost of all vehicles requests $\mathcal{R}$ for a given time horizon T.

$$C_{\gamma_i}(t) = \sum_{i\in\mathcal{R}} C^I_{\gamma_i}(t) + C^T_{\gamma_i}(t) + C^{IB}_{\gamma_i}(t) + \mathfrak{f}^i_b(t) \cdot \left(C^U_{\gamma_i}(t) + C^S_{\gamma_i}(t)\right), \tag{27}$$

$$C_{all} = \sum_{t\in T}\sum_{i\in\mathcal{R}} C_{\gamma_i}(t). \tag{28}$$

III. Problem Formulation

In this section, a critical update threshold $\mathfrak{f}^{i,j}_t$ of jth BVVS is first introduced to trade-off the backup synchronization cost and the BVVS migration cost, and to decide whether the backup is updated, migrated, or skipped. Then, we present the definition and NP-hardness of the problem, and formulate it as Integer Linear Programming.

The specific definition of $\mathfrak{f}^{i,j}_t$ is as follows, and it will be used in Algorithm 3.

Definition 1: Given the deployment locations of VVS and BVVS, VVS computing resources, and unit cost on computing and communication costs, the critical update threshold $\mathfrak{f}^{i,j}_t$ of a BVVS of a vehicle car_i is the frequency at which the sum of the backup updating cost and synchronization cost equals the BVVS migration cost, namely $\mathfrak{f}^{i,j}_t(C^U_{ib_j} + C^S_{ib_j}) = C^I_{ib_j}$. The $\mathfrak{f}^{i,j}_t$ is inversely proportional to the communication distance, and its specific calculation process is as follows:

$$\begin{aligned}\mathfrak{f}^{i,j}_t &= \frac{C^I_{ib_j}}{C^U_{ib_j} + C^S_{ib_j}} \\ &= \frac{c_c \cdot Cap_{f_i}}{\alpha \cdot c_c \cdot Cap_{f_i} + \sum_{e\in P_{v,vb_j}} c_e \cdot e \cdot \alpha \cdot Cap_{f_i}} \\ &= \frac{c_c}{\alpha \cdot c_c + \sum_{e\in P_{v,vb_j}} c_e \cdot e \cdot \alpha} \\ &= \frac{1}{\alpha \cdot (1 + c_e/c_c * |P_{v,vb_j}|)}.\end{aligned} \tag{29}$$

Definition 2: Given a finite time horizon T and a resource-constrained EVM network $G = (V, E)$ that consists of a group V_a of access points co-located with edge server V_e having Cap_{ve} computing resources, a set of mobile vehicle V_c whose trajectory is unknown, and a collection of VVS-enabled metaverse service request $\mathcal{R} = \bigcup_{i=1}^{|V_c|}\{\gamma_i\}$ issued by above each vehicle, the immersive VVS-enabled metaverse services provisioning problem (IVMSP) is to maximize the number of acceptances of metaverse service requests while minimizing their admission cost, through migrating VVSs and BVVSs or adjusting backup updating frequency, subjected to the computing resource limitation on each edge server, where a request is accepted if its various demands such as reliability, AoBI, metaverse service type are met.

Theorem 1: The immersive VVS-enabled metaverse service provisioning problem in a resource-constrained EVM is NP-hard.

Proof: The proof can be found in Appendix A. □

The long-term IVMSP problem is formulated as an integer linear programming problem, where $x_{i,v,u}(t)$, $y^i_{b_j,v,u}(t)$, and $y^i_{b_j,v,vb_j}(t)$ are binary decision variables. In addition, let $p_p = -\log\rho_p$, $p_b = -\log\rho_b$, and $p_{R_i} = -\log R_i$ to make the reliability constraint (formulas (5) and (6)) conform to standard form of integer linear programming.

The problem optimization objective is to

$$\text{Minimize}\quad C_{all} = \sum_{t\in T}\sum_{\gamma_i\in\mathcal{R}} C_{\gamma_i}(t). \tag{30}$$

Subject to:

$$\sum_{u\in V_e} x_{i,v,u}(t) = 1, \gamma_i \in \mathcal{R} \tag{31}$$

$$\sum_{j\in k}\sum_{u\in V_e\setminus v} y^i_{b_j,v,u}(t) = \mathbb{M}(t), \gamma_i \in \mathcal{R} \tag{32}$$

$$\sum_{j\in k}\sum_{vb_j\in V_e\setminus v} y^i_{b_j,v,vb_j}(t) = \mathbb{N}(t), \gamma_i \in \mathcal{R} \tag{33}$$

$$p_p + \sum_{j\in k} p_b \cdot y^i_{b_j,v,vb_j}(t) \le p_{R_i}, \gamma_i \in \mathcal{R} \tag{34}$$

$$\Delta^i_{max}(t)(1 - \mathbb{N}(t) - \mathbb{M}(t)) \le \theta, \gamma_i \in \mathcal{R} \tag{35}$$

$$\sum_{\gamma_i \in \mathcal{R}} Cap_{f_i} \left(x_{i,v,u}(t) + \sum_{j \in k} y^i_{b_j,v,u}(t) \right) \leq Cap_{ve_u}(t), \forall u \in V_e \tag{36}$$

$$x_{i,v,u}(t) \in \{0,1\}, u \in V_e, \gamma_i \in \mathcal{R} \tag{37}$$

$$y^i_{b_j,v,u}(t) \in \{0,1\}, u \in V_e, \gamma_i \in \mathcal{R} \tag{38}$$

$$y^i_{b_j,v,vb_j}(t) \in \{0,1\}, vb_j \in V_e, \gamma_i \in \mathcal{R} \tag{39}$$

$$y^i_{b_j,v,u}(t) + y^i_{b_j,v,vb_j}(t) \leq 1, v \in V_e, j \in k, \gamma_i \in \mathcal{R} \tag{40}$$

The formula (30) is the minimization objective of the IVMSP problem. Constraint (31) indicates that the VVS required by a mobile vehicle is migrated to only one server per time slot. Constraints (32) and (33) indicate that $\mathbb{M}(t)$ or $\mathbb{N}(t)$ backups need to be migrated or synchronized at each time slot, respectively. Constraint (34) represents the minimum number of backups required by each vehicle. Constraints (34) and (35) ensure that user requirements on reliability and age of backup information must be satisfied. Inequality (36) indicates that resources occupied by all VVS and BVVS in an edge server do not exceed the remaining computing capacity of the server. Inequality (40) represents that at most one mechanism can be selected for each backup per car per time slot, namely migration backup, adjustment update frequency, or skip.

IV. Metaverse Service Home Prediction Algorithm

To solve the IVMSP problem, this paper designs a comprehensive and immersive resource explorer, named spotlighter, which is composed of a metaverse service home prediction algorithm, a VVS migration algorithm, and a cost-adaptive AoBI guarantee algorithm, as shown in Fig. 1.

In this section, a deep-learning metaverse service home prediction algorithm based on the transformer is designed to predict the next service ownership, namely the connected access point in the next slot, by combining the current and historical information of each vehicle.

A. Model Architecture

The architecture of the prediction model in Fig. 3 consists of input data, embedding, and modeling, which are described as follows.

Input data: A set of tuples $D = (X, Y)$ is fed to the model as input data, where X is the historical information and Y is the ground-truth information in the next slot for all cars.

In terms of historical information of each vehicle car_i, in addition to the attributes $AP_i(t)$ and $S_i(t)$, three auxiliary variables are introduced to compose an input data tuple $X_i(AP_i(t), AP_i\_a(t), Tra_i\_a(t), LD_i(t), S_i(t))$. The $AP_i\_a(t)$ represents the angle between the connected server $AP_i(t)$ and the car_i, and the $Tra_i\_a(t)$ indicates the angle between the past and current location of the car_i at $t-1$ and t slot. The $LD_i(t)$ is the location distance gap between the $AP_i(t)$ and the car_i.

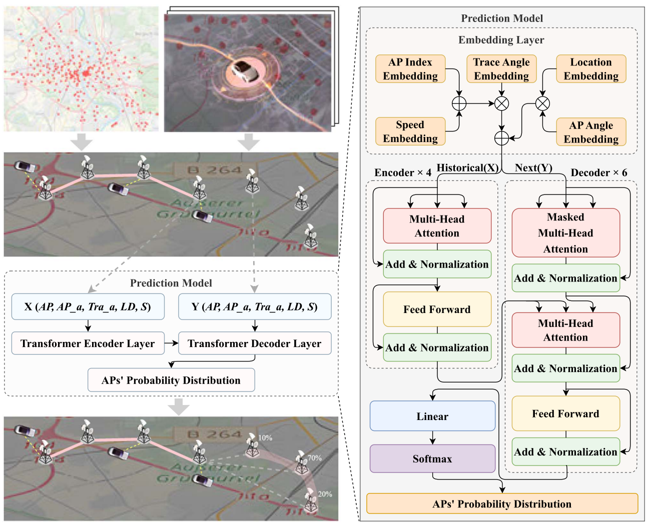


Fig. 3. The architecture of the prediction model.

In terms of ground-truth information of each vehicle car_i, in addition to five attributes in X_i, two tokens $begin$ and end are introduced to mark the beginning and end position of $Y_i(AP_i(t+1), AP_i\_a(t+1), Tra_i\_a(t+1), LD_i(t+1), S_i(t+1))$. Note that the $begin$ token with Y_i is input into the decoder when model training, only the $begin$ token is passed to when model inference. The end token is just used to mark the end position of Y_i in the model output.

Embedding: Having the raw data X_i been transformed into corresponding multiple $N_\mathfrak{e}$-dimensional vectors, each vector is an output of an exclusive embedding layer with only one fully-connected neural network. These vectors then are used to obtain an adequate representation $\mathfrak{E}_{fin_i}$ of car_i, which is calculated as follows:

$$\mathfrak{E}_{fin_i} = (\mathfrak{e}_{ap} + \mathfrak{e}_s) * \mathfrak{e}_{tra\_angle} + \mathfrak{e}_l * \mathfrak{e}_{ap\_angle}, \tag{41}$$

where $\mathfrak{e}_{ap}$ and $\mathfrak{e}_s$ are AP index and car speed embedding, respectively. $\mathfrak{e}_{ap\_angle}$ indicates the AP angle embedding, $\mathfrak{e}_{tra\_angle}$ is trace angle embedding, and $\mathfrak{e}_l$ represents location embedding. The $\mathfrak{E}_{fin}$ has a certain degree of interpretability during embedding, which is the shine and special place compared to other algorithms. It combines the departure information $\mathfrak{e}_s * \mathfrak{e}_{tra\_angle} + \mathfrak{e}_l * \mathfrak{e}_{ap\_angle}$ and the direction information $\mathfrak{e}_{ap} * \mathfrak{e}_{tra\_angle}$, that is, the separation distance between an old service home and the car, and the connection direction of a new service home.

After the embedding process, the composite representation $\mathfrak{E}_{fin_i}$ of every vehicle is fed into the transformer model. In addition, the ground-truth information Y_i also needs to go through an embedding process together with $begin$ token to become $\mathfrak{E}^y_{fin_i}$ before entering the model.

Modeling: The prediction model of metaverse service home is an encoder-decoder model [15], where the input of the encoder is the historical information vector $\mathfrak{E}_{fin_i}$, and that of the decoder is corresponding vector $\mathfrak{E}^y_{fin_i}$. Both the encoder and decoder are composed of two blocks, namely a multi-head attention block and a feed-forward block. The multi-head attention block includes multiple attention heads, which can assign different weights to a set of time slots that we focus on. In addition, the first layer of the decoder is a multi-head attention block with a mask, and it can allow attention heads to focus on different lengths of time slots based on different masks. The feed-forward block consists of two fully-connected neural networks and a $ReLU$ activation function, which performs linear and nonlinear transformations on the previous layer. After each block, residual connections and layer normalization are used to calculate the block's output. As shown in Fig. 3, embedding vectors $\mathfrak{E}_{fin_i}$ and $\mathfrak{E}^y_{fin_i}$ pass through 4 encoder layers and 6 decoder layers (above parameters have better performance), and then sent to a single-layer fully-connected neural network with $softmax$ activation function, where the probability distribution $\hat{Y}_i^{ap}$ of all access points $AP_i(t+1)$ to be connected for each vehicle in the next slot is calculated.

B. Training Process

The training process of the metaverse service home prediction model is shown in Algorithm 1, described as follows.

Step 1: Training data set D is initialized as an empty set, then fill with a set of X_i and Y_i (lines 2-4). The history information H_i of each vehicle car_i is split into $\lfloor \frac{lh_i}{rl+1} \rfloor$ blocks, and stored in D, where lh_i is the length of H_i and rl is a given retrieve length. In each block, only the $rl+1$-th data is used as Y_i, and the rest are taken as a set of X_i.

Step 2: Training data set D is split into $\lfloor \frac{ld}{batch} \rfloor$ chunks, where ld and $batch$ are the length of D and chunk TS_{iter}, respectively. In each iteration $iter$, $|batch|$ vehicle information D_i are randomly selected and packet a chunk TS_{iter} as model input, and delete TS_{iter} from D (lines 7-8).

Step 3: Update model parameters by gradient descent method with the aim of minimizing the average cross entropy loss function $Loss(\bar{Y}_{iter})$ (line 9), which is defined as follows:

$$CE(Y_i^{ap}, \hat{Y}_i^{ap}) = -\sum_{1\leq j\leq |AP|+2} Y_{i,j}^{ap} \cdot \log \hat{Y}_{i,j}^{ap},$$

$$Loss(\bar{Y}_{iter}) = \frac{1}{|batch|} \sum_{1\leq i\leq |batch|} CE(Y_i^{ap}, \hat{Y}_i^{ap}),$$

where Y_i^{ap} is a real AP index with one-hot encoding, and $\hat{Y}_i^{ap}$ is predicted vectors. The $\hat{Y}_i^{ap}$ represents a set of probabilities $\hat{Y}_{i,j}^{ap}$ that a car is about to connect to access point AP_j, which has the same dimension $|AP|+2$ as Y_i^{ap}, representing AP number, $begin$, and end tokens. Finally, the average loss $Loss(\bar{Y}_{iter})$ of all $Y_i^{ap} \in TS_{iter}$ is calculated in each iteration.

Algorithm 1: Training of Prediction Model.

```
Input: The historical data $\bigcup_{i\in n} H_i$ of $n$ vehicles.
Output: A well-trained prediction model.
1  begin
2      $D \leftarrow \emptyset$, $TS \leftarrow \emptyset$, $round \leftarrow 0$, $iter \leftarrow 0$;
3      for each historical data $H_i$ of a vehicle $car_i$ do
4          $H_i$ is split into $\lfloor \frac{lh_i}{rl+1} \rfloor$ blocks, and stored in $D$;
5      while $round < epoch$ do
6          while $iter < \lfloor \frac{ld}{batch} \rfloor$ do
7              Sample $|batch|$ $D_i \in D$ into $TS_{iter}$;
8              $D \leftarrow D \backslash TS_{iter}$;
9              Update model parameter to minimize the
                 average loss $Loss(\bar{Y}_{iter})$;
10             $iter \leftarrow iter + batch$
11         $round \leftarrow round + 1$;
```

V. VVS Migration Algorithm

In this section, we start by explaining the core idea and each step of the randomized algorithm `Algorithm` 2, and then analyze its approximation ratio and time complexity.

A. VVS Migration Algorithm

VVS migration algorithm is based on randomized rounding and integer linear programming technologies, which aims to enable the required VVSs to migrate with moving vehicles at the lowest migration cost.

The VVS migration subproblem in IVMSP is first modeled as the following integer linear programming.

$$\text{Minimize} \quad \sum_{t\in T} \sum_{\gamma_i \in \mathcal{R}} C^I_{\gamma_i}(t) + C^T_{\gamma_i}(t).$$

Subject to:

$$(31), (36), (37).$$

A randomized algorithm `Algorithm` 2 for the VVS migration subproblem is described as follows.

Step 1: Let $M\mathcal{R}(t)$ be the set of requests to be migrated at time slot t (lines 3-5). The new service home $\hat{AP}_i$, the closest access point for a car, is first calculated by the model of Algorithm 1. It is a continuous prediction process, this is, $\hat{AP}_i$ at t time slot is predicted by the predicted results (old service home) at $t-1$ time slot (line 3). Then, the location consistency between $\hat{AP}_i$ and AP_i is used to determine whether the service needs to be migrated (lines 4–5).

Step 2: An optimal solution $\widetilde{OPT_{lp}}$ on the relaxed version of the VVS migration subproblem is obtained (lines 6–7). For each request in $M\mathcal{R}(t)$, a binary variable $x_{i,v,u}$ is converted to continuous one $\widetilde{x}_{i,v,u}$ to improve solver's efficiency and quickly obtain the lowest cost $\widetilde{OPT_{lp}}$.

Step 3: A suboptimal solution OPT is obtained with high probability $\min\{1-1/|\mathcal{R}|, 1-1/|2V|\}$, delivered by a randomized rounding technique (lines 8–15). Inspired by the idea of the roulette wheel selection, the selection probability of an edge server V_{e_u} increases as its value $\widetilde{x}_{i,v,u}$ increases. Specifically, Algorithm 2 samples the threshold of selection probability ζ uniformly from $[0,1]$, and set cumulative probability sum to 0 (lines 9–10). To ensure deployment location constraints, each VVS is only deployed to an edge server. Only the edge server that satisfies the $sum \geq \zeta$ condition first will be selected, namely, $x_{i,v,u'_j} = 1$ (lines 13–15).

Algorithm 2: Training of Prediction Model.

```
Input: An EVM network $G(t-1)$, a group of mobile cars, and a set $\mathcal{R}(t-1)$ of VVS-enabled metaverse request issued by cars at $t-1$ slot.
Output: A service migration policy in $t$ slot, that is, a service redeployment policy in $t$ slot
1  begin
2      for each vehicle $car_i \in \mathcal{R}$ do
3          Predict new service home $\hat{AP_i}$ in $t$ slot based on old service home $AP_i$ already predicted in $t-1$ slot by algorithm 1's model;
4          if $AP_i \neq \hat{AP_i}$ then
5              $M\mathcal{R}(t) \leftarrow \gamma_i$;
6      For $\gamma_i \in M\mathcal{R}(t)$, the discrete variables $x_{i,v,u}$ are relaxed into the continuous variables $\widetilde{x}_{i,v,u}$;
7      The optimal solution $\widetilde{OPT_{lp}}$ is obtained by linear programming solver;
8      for each vehicle $car_i \in M\mathcal{R}(t)$ do
9          Generate a random number $\zeta \in [0,1]$;
10         $sum \leftarrow 0$;
11         for each edge server $v_{u_j} \in V_e$ do
12             $sum \leftarrow sum + \widetilde{x}_{i,v,u_j}$;
13             if $sum \geq \zeta$, when $v_{u_j} = v_{u'_j}$ then
14                 $x_{i,v,u} \leftarrow 1, u = u'_j$;
15                 $x_{i,v,u} \leftarrow 0, u \neq u'_j$;
16     return A suboptimal solution $OPT = \cup x_{i,v,u}$, and a set $M\mathcal{R}(t)$ of migration requests.
```

B. Algorithm Analysis

Theorem 2: Given a resource-constrained EVM network $G = (V, E)$, a set of mobile vehicle V_c whose trajectory is unknown, and a collection $\mathcal{R}$ of VVS-enabled metaverse service request γ issued by each vehicle, there is a randomized algorithm, `Algorithm` 2, with high probability $\min\{1-1/|\mathcal{R}|, 1-1/|2V_e|\}$, and the approximation ratio of the algorithm is 2, provided that $OPT \geq 6\Phi(1-1/|\mathcal{R}|)$, $Cap_v \geq \lambda^{max} Cap_f^{max} \ln V_e$ and the computing consumption of any edge server is no more than treble of its capacity. The algorithm takes $O(|V_e| \cdot |M\mathcal{R}|)$, where $|V_e|$ is the number of access points, Cap_f^{max} indicates the maximum amount of computing resource demanded by any single VVS among all VVSs, where $|\mathcal{R}|$ is the number of requests, and Φ is the maximum traffic migration cost of any shortest path among all shortest paths in V_e or the maximum VVS re-instantiation cost of any VVS among all VVSs.

Proof: The proof can be found in Appendix B. □

VI. Cost-Adaptive AoBI Guarantee Algorithm

In this section, we deal with AoBI problem, and start with the key idea of an algorithm for it, and then detail each step of the proposed algorithm. We finally analyze the performance ratio of the proposed algorithm and its time complexity.

Algorithm 3: The Cost-Adaptive AoBI Guarantee Algorithm.

```
Input: An EVM network $G(t)$, a set of requests $\mathcal{R}(t)$, and a set of migration requests $M\mathcal{R}(t)$ at slot $t$.
Output: A adjustment policy of all backups in slot $t$.
1  begin
2      for each vehicle $car_i \in \mathcal{R}(t)$ do
3          if $t = 1$ then
4              Calculate the current reliability of all the VVS and the number $k_i$ of backups required based on formulas (5) and (6);
5              Deploy the $k_i$ backups on the Top-$k_i$ servers closest to the corresponding VVS;
6              Calculate the initial update frequency $\mathfrak{f}_b^i(1)$ for the $k$ backups based on formulas (18) and the age threshold $\theta$;
7          else
8              $\xi \leftarrow 0$;
9              for each backup $f_{ib_j}$, $j \in \{1, \ldots, k_i\}$ do
10                 if an AoBI violation occurred then
11                     Calculate the expected frequency $\mathfrak{f}_b^{i,j}(t)$ based on formulas (17);
12                     if $\mathfrak{f}_b^{i,j}(t) > \mathfrak{f}_{max}$ or not exist then
13                         Migrate $f_{ib_j}$ from edge server $V_{e_v}$ to $V_{e_u}$ closest to the car, and update $y_{ib_j,v,u}(t)$;
14                     else
15                         Calculate the critical threshold $\mathfrak{f}_t^{i,j}$ based on the formula (29);
16                         if $\mathfrak{f}_b^{i,j}(t) \leq \mathfrak{f}_t^{i,j}$ then
17                             Use $\mathfrak{f}_b^{i,j}(t)$ as the new frequency for $j$th backup;
18                             if $\mathfrak{f}_b^{i,j}(t) \geq \xi$ then
19                                 $\xi \leftarrow \mathfrak{f}_b^{i,j}(t)$;
20                         else
21                             Migrate $f_{ib_j}$ from server $V_{e_v}$ to $V_{e_u}$ closest to the car, and update $y_{ib_j,v,u}(t)$;
22             Update $\mathfrak{f}_b^i(t) \leftarrow \xi$;
23     return Modified $\mathfrak{f}_b^i(t)$ and backup location $y_{ib_j,v,u}$
```

A. Cost-Adaptive AoBI Guaranteed Algorithm

The cost-adaptive AoBI guarantee algorithm is first proposed to ensure the ages of all backup information by selecting migration or adjusting update frequency adaptively. Then, the critical update threshold $\mathfrak{f}_t^{i,j}$ of each BVVS is introduced to strive for a nice tradeoff between the BVVS synchronization cost and the BVVS migration cost. Next, the implementation of the cost-adaptive AoBI guaranteed algorithm (Algorithm 3) is described as follows.

Step 1: At the beginning of the algorithm, the setting for the appropriate number of backups for each user is calculated by formulas (5) and (6) iteratively. Then, the k_i backup deployment locations of vehicle car_i are selected based on the distance between the vehicle location $L_i(t)$ and edge server locations in G, and the backup update frequency of each VVS-enabled metaverse request is calculated by using formula (18) (lines 4–6).

Step 2: For each backup, the update frequency mechanism is activated first when an AoBI violation is detected. Specifically, the lazy update strategy is used during the AoBI-guaranteed process, which means that a set of update packets is sent only at the latest time before AoBI is violated. Therefore, without violating AoBI, both the update frequency and the synchronization cost decrease as the waiting time $T^W_{i,j}$ for updates increases. The expected update frequency $\mathfrak{f}^{i,j}_b(t)$ is calculated to avoid the violation (line 11), which is the lowest update frequency $\mathfrak{f}^{i,j^-}_b(t)$ for $\mathfrak{f}^{i,j}_b(t) \in [1/(\theta - T^T_{i,j}), 1/T^T_{i,j}]$. When the frequency $\mathfrak{f}^{i,j}_b(t)$ does not exist or is greater than the maximum frequency $\mathfrak{f}_{max}$. The backup can only be migrated to the edge server closest to the car and its new migration location $y_{ib_j,v,u}(t)$ is also updated (lines 12-13). Otherwise, the algorithm enters the cost-adaptive AoBI adjustment phase, which is detailed as follows.

Step 3: The critical update threshold $\mathfrak{f}^{i,j}_t$ is introduced to help adaptively decide whether to migrate a backup or adjust update frequency (line 15). The actual meaning of $\mathfrak{f}^{i,j}_t$ is that when the frequency is adjusted in the range $[0, \mathfrak{f}^{i,j}_t]$, the update and synchronization cost of BVVS is always less than its migration cost. If $\mathfrak{f}^{i,j}_b(t)$ is less than $\mathfrak{f}^{i,j}_t$, the sum of the update and the synchronization cost is less than the BVVS migration cost, so the updating frequency is increased to $\mathfrak{f}^{i,j}_b(t)$ (lines 16-17); Otherwise, the backup is migrated from the original hosting server V_{e_v} to edge server V_{e_u} that is closest to the user, where the distance between them is the shortest path in G, and its new migration location $y_{ib_j,v,u}(t)$ is also updated.

Step 4: The VVS sends update packets to its corresponding backups in the form of a broadcast (line 22). The actual update frequency $\mathfrak{f}^i_b$ is the lower bound $\mathfrak{f}^{i,j'^-}_b(t)$ of the j'th backup with the longest synchronization distance among all backups. Specifically, the maximum value ξ among all backup update frequencies $\mathfrak{f}^{i,j}_b(t)$ is used as the synchronous frequency of VVS, and when the number of packets received by each BVVS exceeds $\mathfrak{f}^{i,j}_b(t)$, the excess packets are discarded.

B. Algorithm Analysis

Theorem 3: Given a VVS-enabled metaverse request γ_i with AoBI demand θ, a set $\mathcal{B}_i$ of corresponding BVVSs f_{ib_j}, and multiple synchronization paths P_{v,vb_j} from VVS to each backup $f_{ib_j} \in \mathcal{B}_i$. For each VVS f_i, the update frequency $\mathfrak{f}^i_b$ of backup information has lower and upper bounds, $\mathfrak{f}^i_b \in [1/(\theta - T^T_{i,j'}), 1/T^T_{i,j'}]$, which is only determined by a special BVVS $f_{ib_{j'}}$, where $T^T_{i,j'}$ is the transmission time of the j'th BVVS with the longest synchronization distance among all backups.

Proof: The proof can be found in Appendix C. □

Corollary 1: The AoBI requirement limits not only backup freshness but also transmission latency, that is, given an AoBI value θ, there is an upper bound $T^{T+}_i = \theta/2$ of transmission time for any backup $f_{ib_j} \in \mathcal{B}_i$.

Proof: Based on Theorem 3, there is an available range of update frequency for each BVVS

$$\frac{1}{\theta - T^T_{i,j}} \leq \mathfrak{f}^{i,j}_b \leq \frac{1}{T^T_{i,j}}.$$

Then, we have

$$\frac{1}{\theta - T^T_{i,j}} \leq \frac{1}{T^T_{i,j}} \Rightarrow T^T_{i,j} \leq \frac{\theta}{2}. \tag{42}$$

The theorem then follows. □

Theorem 4: Given a resource-constrained EVM network $G = (V, E)$ and a set $M\mathcal{R}(t)$ of migration requests issued by mobile vehicles, there is a cost-adaptive AoBI guaranteed algorithm with the worst-case performance bound $2\theta\Xi$ to adjust AoBI of all backups in $O(O(|V_e||\mathcal{R}||k_{max}+1|))$, where C is the total sum of synchronization cost obtained by Algorithm 3, C^* is the optimal cost of the optimal scheme, f_{max} and $|k_{max}|$ is the maximum update frequency and BVVS numbers among all requests, $|p|$ is the average number of backups to ensure AoBI by using the update frequency policy, and $\Xi = \mathfrak{f}_{max}|k_{max}|/|p|$.

Proof: The proof can be found in Appendix D. □

VII. Performance Evaluation

In this section, we first conduct simulation experiments using real data sets, and then evaluate the performance of the designed algorithms for the immersive VVS-enabled metaverse service provisioning problem. Finally, we study key parameters that affect the performance of the proposed algorithms.

A. Experimental Environment Setting

In terms of network topology, we simulate an EVM network $G = (V, E)$ by using real-world 246 locations of access points and 700000 mobility traces of vehicles in Cologne city [16]. The EVM network is a subgraph extracted from the above real dataset, which includes all access points and 250 vehicles. 70% of the network size access points are randomly selected to coexist with edge servers, and the computing capacity of each server ranges from 4000 to 8000 MHz [8]. The unit prices for computing resources c_c and links c_e are \$0.25 per MHz and \$0.0035 per $Mbps$, respectively, which are the official Amazon Cloud prices [12], [17]. In addition, the type number $|F|$ of VVS-enabled metaverse services in the EVM is set to 40, each service consumes roughly 200 to 400 MHz computing resources, and its reliability ranges from 0.7 to 0.8.

In terms of mobile vehicles, each one car_i issues a VVS-enabled metaverse service γ_i with various service demands. Specifically, the reliability demand R_i is set from 0.8 to 1, the accuracy demand λ_i ranges from 0.9 to 1, and the AoBI demand θ is set to 5 seconds [18], [19]. The proportional coefficient α between the size of backup status information and the VVS computing consumption is set to 0.1. The data rate B_i of vehicles is set to 24.02 $Mbps$ which is the average download rate of users accessing the Internet in 2020 [20]. The maximum update frequency $\mathfrak{f}_{max}$ is 4 times per second.

TABLE II
THE PARAMETER OF THE PREDICTION MODEL

	batch	Encoder layer	Decoder layer	Attention head	rl	N_e	Feed-forward output	Linear output
Size	128	4	6	32	1	1024	2048	248

In terms of the prediction model, we randomly select 70% of datasets in mobility traces of vehicles in Cologne city [16] for training and 30% of datasets for testing. We also set the learning rate to $5e-5$ at the beginning, and set an exponential decay coefficient of the learning rate to 0.9 per epoch. The specific parameters of the model are shown in Table II. The linear output is a vector of length 248, derived from the sum of 246 and 2 where 246 represents the number of base stations and 2 indicates two tokens $begin$ and end.

The above values are default values for each trial unless otherwise specified, and all experiments were conducted on a personal machine with a 2.30 GHz Intel(R) Core(TM) I7-10875 CPU and 16 GB of RAM.

B. Comparison Algorithms

To evaluate the performance of the proposed algorithms, the following comparison algorithms are proposed.

Completely Random (CR): It first randomly selects the migration location for each VVS. Then, when the corresponding BVVSs have AoBI conflict, migration locations of BVVSs are also randomly decided or the corresponding update frequency of each backup is generated.

Greedy [21] + Only Frequency [11] [22] (GOF): It first uses a greedy algorithm with an approximate ratio to ensure that a VVS moves with a vehicle at a lower cost. Then, it takes the common approach in existing works in the survey [23] to ensure AoBI, which is to adjust only the backup update frequency when an AoBI conflict occurs.

Greedy [21] + Only Migration (GOM): The first stage of this algorithm is the same as that of the GOF algorithm. In the second stage, it only migrates BVVS near the corresponding VVS to avoid AoBI conflicts, and the update frequency of each BVVS under the migrated location is calculated.

ILP [24] + Asynchronization (ILPA): This algorithm knows the true locations of each vehicle instead of using predictive outcomes. In addition, it first solves the integer linear programming subproblem modeled in Section IV by adopting the DOcplex solver in Python, and then, an asynchronous update strategy (each backup can use its own update frequency) is introduced to ensure the AoBI, where an anti-perturbation factor with a value of 0.001 is added to the update frequency of each backup. This value can reduce the sensitivity between the number of the frequency and dynamic transmission delay by setting a slightly higher update frequency in advance.

Note that according to the oneness principle of control variates, we regard **ILPA_pre**, the variant of the applied prediction results, as *the optimal strategy*, whereas the ILPA is only treated as an actual solution.

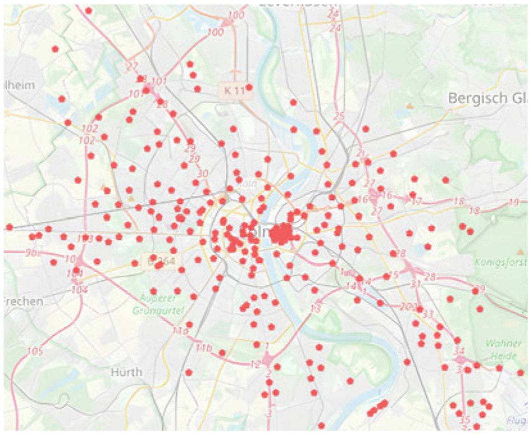
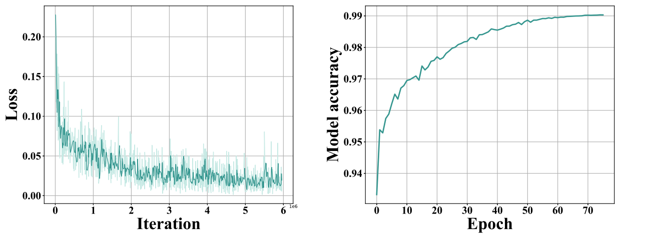


Fig. 4. The locations of access points in Cologne city, Germany [16].



(a) Convergence of loss function (b) Convergence of model accuracy

Fig. 5. Performance convergence of metaverse service home prediction algorithm.

TABLE III
COMPARISON OF PREDICTION ACCURACY FOR DIFFERENT ALGORITHMS IN [25]

Model	CNN-LSTM	CNN	Seq2Seq	Dilated Causal CNN	**Transformer (our)**
Accuracy	93.85%	94.98%	95.1%	97.22%	**99.02%**

TABLE IV
PARAMETER SETTINGS

	$\|V_e\|$	T	r	θ	λ_i	$\|\mathcal{R}\|$	Cap_f	$\|F\|$
Set 1	$40\% \sim 80\%$	10	$0.7 \sim 0.8$	5	$90\% \sim 100\%$	$200 \sim 400$	40	150
Set 2	70%	$1 \sim 30$	$0.7 \sim 0.8$	5	$90\% \sim 100\%$	$200 \sim 400$	40	150
Set 3	70%	10	$0.675 \sim 0.875$	5	$90\% \sim 100\%$	$200 \sim 400$	40	150
Set 4	70%	10	$0.7 \sim 0.8$	$3 \sim 7$	$90\% \sim 100\%$	$200 \sim 400$	40	150
Set 5	70%	10	$0.7 \sim 0.8$	5	$72.5\% \sim 92.5\%$	$200 \sim 400$	40	150
Set 6	70%	10	$0.7 \sim 0.8$	5	$90\% \sim 100\%$	$220 \sim 380$	40	150
Set 7	70%	10	$0.7 \sim 0.8$	5	$90\% \sim 100\%$	$200 \sim 400$	$20 \sim 40$	150
Set 8	70%	10	$0.7 \sim 0.8$	5	$90\% \sim 100\%$	$200 \sim 400$	40	$50 \sim 250$

C. Algorithm Performance Evaluation

We study the effect of iteration numbers on algorithm convergence by varying it from 1 to 6000000. Fig. 5(a) shows that the algorithm converges between 0.05 and 0.001 with the increase in the number of iterations. It can also be concluded from Fig. 5(b) that the model accuracy is improved from 93.32% to 99.02% after 76 training epochs in advance on the offline base set. Compared with previous prediction algorithms in [25], the accuracy of the proposed algorithm is improved by 1.8 % as shown in Table III.

The influence of 8 key parameters on other algorithm performance is verified, and the specific experimental sets are shown in Table IV. The basic idea of parameter configuration is as

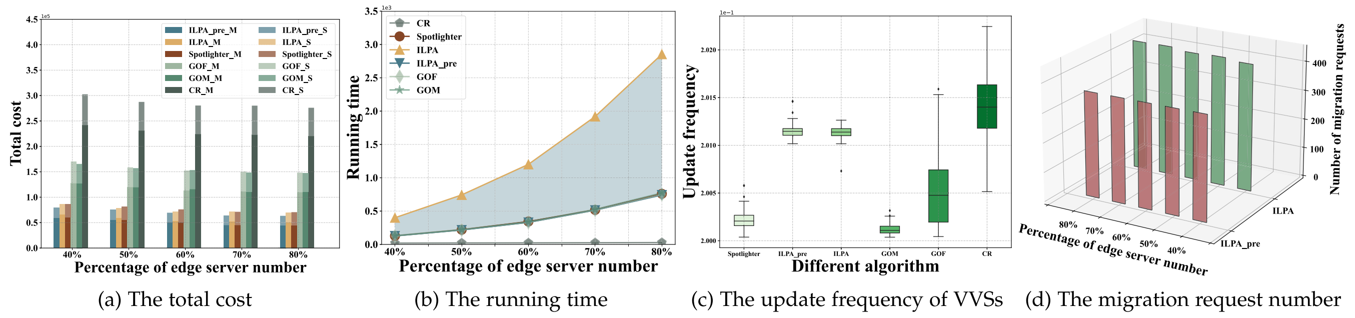

(a) The total cost (b) The running time (c) The update frequency of VVSs (d) The migration request number

Fig. 6. Performance of different algorithms by varying the percentage of numbers of edge servers from 40% to 80%.

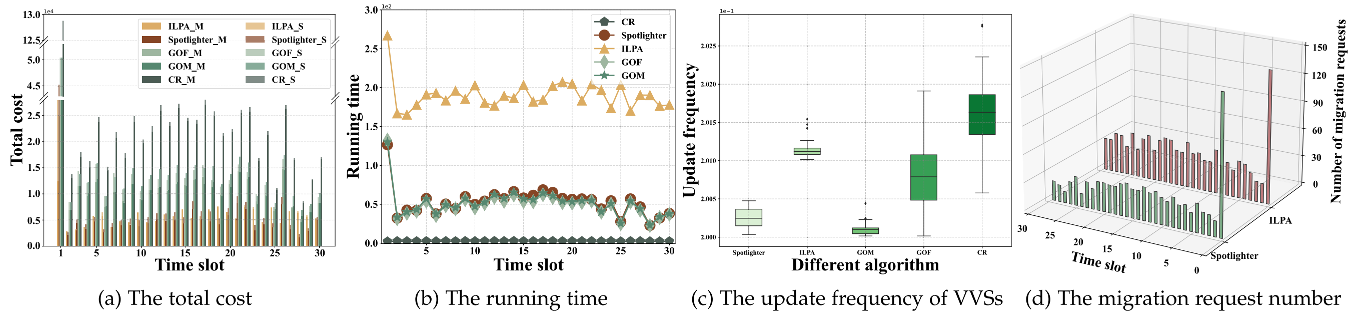

(a) The total cost (b) The running time (c) The update frequency of VVSs (d) The migration request number

Fig. 7. Performance of different algorithms by varying the number of time slots from 1 to 30.

follows: The range of computing capacity of each server and VVS is adopted from existing work, such as the ones from [8] and [26]. Based on this, the maximum throughput of the network is calculated, which limits the number of arrival requests at each time slot. Also, different service requirements and VVS attributes are determined according to application scenarios. Each type of parameter changes from small to large, which has universal applicability because this progressive parameter setting can reflect the impact of the parameter on algorithmic performance.

1) Impact of the Number of Edge Servers on Performance of Algorithms: We first adopted the parameter setting **set 1** in Table IV, which increases the percentage of the number of edge servers from 40% to 80% of the network size, namely improving the number and density of edge servers in the EVM, to study the performance impact. From Fig. 6(a), the total acceptance cost of *ILPA_pre*, consisting of the migration cost *ILPA_pre_M* and the synchronization cost *ILPA_pre_S*, is the lowest one for all algorithms using predicting results. The acceptance costs of *ILPA* and *spotlighter* are almost the same, which are much smaller than those by the rest of the three algorithms, that is, *spotlighter* can save 47.6%, 48.2%, and 71.5% of the cost, respectively. According to Fig. 6(b), the running times of all algorithms increase with the increase in the number of edge servers. In addition, it can be seen that *ILPA_pre* can save at least 312.4% of the running time by sacrificing less than 11.7% of the cost error compared with *ILPA*, using real vehicle location data. Fig. 6(c) illustrates that the distribution of VVSs' average update frequency of *spotlighter* is between *GOM* and *GOF*, lower than those of the *ILPA* and *ILPA_pre*. This is because *spotlighter* adaptively weighs in on migration costs and update frequency costs, rather than carelessly making choices. Fig. 6(d)shows that the number of migrations based on the metaverse service home prediction algorithm is lower than that of the algorithm of using real vehicle location due to deviations from continuous prediction, which is why the cost of *ILPA_pre* is lower than that of *ILPA* in Fig. 6(a).

2) Impact of the Time Horizon Length on Performance of Algorithms: We then employed the parameter setting **set 2** in Table IV, where the length of the time horizon increases from 1 to 30, to investigate its impact on algorithmic performance. Fig. 7(a) shows that the total costs of *spotlighter* and *ILPA* are the lowest ones among the costs of all algorithms, and the average gap of the total cost between *spotlighter* and *ILPA* is no greater than 21.3% in the first 20 time slots. However, the gap becomes widened after that due to the inevitable accumulation effect of prediction bias in migration numbers as shown in Fig. 7(d). Even if the prediction algorithm can achieve a result with 99.02% accuracy, the prediction bias is still magnified over time (prediction rounds) during continuous predictions, because the new predictions are based on previous prediction results. As shown in Fig. 8, the prediction bias can be divided into the early arrival (19.3%), late arrival (32.7%), and complete deviation (48%) respectively, which causes fluctuations in the number of migration requests, and then affects the total acceptance cost. Regarding the early and late arrival, vehicle trajectories closely resemble the actual path, albeit with a slightly adjusted timetable resulting in either early or delayed migration, while complete deviation causes service interruptions when vehicles are out of communication range.

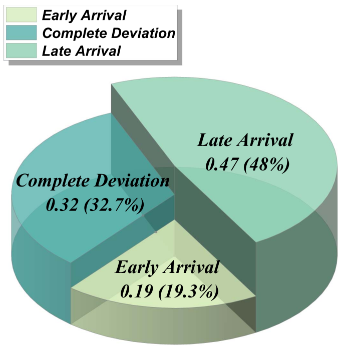
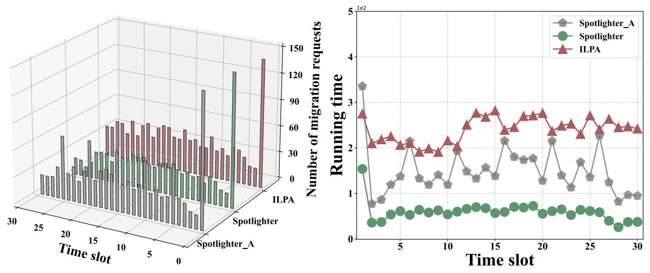


Fig. 8. The distribution (0.98%) of different prediction bias.



(a) The migration request number (b) The running time

Fig. 9. The effect of real location correction on algorithms.

The bias can also be corrected by inserting some real coordinates in the continuous prediction process (e.g., *spotlighter_A* algorithm), but it takes more cost and time in the next time after the correction to bring the predicted and real migration numbers closer, as shown in Fig. 9. Specifically, after the real vehicle coordinates are inserted at 5, 10, 15, 20, and 25 time slots, the number of migration requests in the next slot spikes due to an adjusted operation for the accumulated deviation, yet then it will be closer to the real value. In addition, a clear trend in Fig. 7 is that the different performances of all algorithms fluctuate dynamically over time in a stable region, and the overall performance distribution is the same as in **set 1**. This is because the number of migration requests fluctuates over time due to unknown vehicle trajectories in each time slot.

3) Impact of Different Service Requirements on Performances of Algorithms: Thirdly, the performance impacts caused by different service requirements, including VVS reliabilities, AoBI requirements, and accuracy requirements are investigated, respectively.

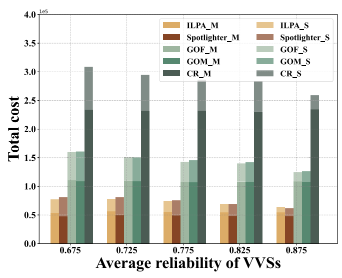
We adopted the parameter setting **set 3** in Table IV, which increases the average reliability of all VVSs from 0.675 to 0.875, to investigate performance impact. As can be seen from Fig. 10(a), the acceptance cost of all algorithms gradually decreases as VVS reliabilities go up, where the cost of *CR* is always the highest, that of *GOF* and *GOM* are in the middle, and *ILPA* and *spotlighter* have the lowest cost. This is due to the fact that the number of backups required per vehicle reduces as VVS reliabilities increase, which results in lower remigration costs and synchronization costs. From Fig. 10(b) and (d), VVS reliabilities have little effect on the running time and the number of migration requests. From Fig. 10(c), the distribution of update frequencies of VVSs is the same as in **set 1**, that of *spotlighter* is the second lowest among all.

We made use of the parameter setting **set 4** in Table IV, which increases the AoBI requirements of all vehicles from 3 to 7, to investigate performance impact. Fig. 11(a) illustrates that the acceptance costs for all algorithms decrease as user tolerance for AoBI grows. This is due to the lazy update strategy adopted by all algorithms, that is, the backup information is updated at the latest. Therefore, the update frequency of all algorithms would gradually fall when AoBI rises, which can also be demonstrated from Fig. 11(d). In addition, the running time and the number of migration requests are basically stable, as shown in Fig. 11(b) and (d).

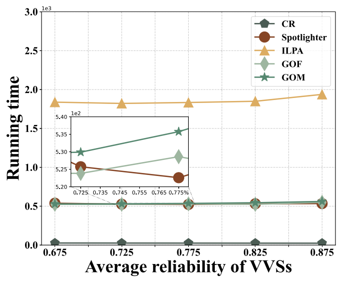
We employed the parameter setting **set 5** in Table IV, which increases the average accuracy requirements of all vehicles from 72.5% to 92.5%, to investigate performance impact. The EVM systems need to deploy corresponding bigger models with larger parameters as the accuracy requirement of each user enhances, thus incurring more migration costs and synchronization costs, as shown by Fig. 12(a). In addition, from Fig. 12, the accuracy has a small effect on different algorithms except for acceptance cost.

4) Impact of Different VVS Attributes on Performance of Algorithms: Fourthly, we adopted the parameter setting **set 6** in Table IV to investigate the impact of different VVS attributes, including VVS computing capacity and VVS type number, on the algorithm performances, by varying the average consumption of computing capacity of VVSs from 220 to 380. We then increased the number of VVS types by employing parameter setting **set 7** in Table IV. From Fig. 13(a), it can be seen that both types of costs rise with the increase on the average computing capacity required by VVS in *ILPA*, *spotlighter*, and *CR*, while only the synchronization costs of *GOF* and *GOM* increase. This is because greedy algorithms always choose a location with the lowest cost at each iteration when migrating services. Consequently, this does not result in an optimal solution, but it has upper bounded the migration cost, preventing it from rising infinitely. From Fig. 13(b), the running times of different algorithms fluctuate within a certain range, due to the random generation of service demand parameters in given ranges during each run. In addition, since the VVS attributes are independent of the locations of vehicles, this has little effect on part performances of different algorithms as shown in Figs. 13(c), (d), and 14.

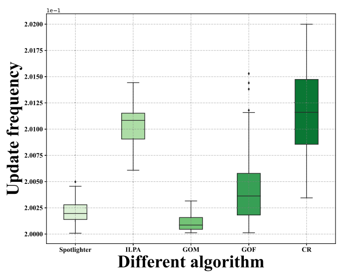
5) Impact of the Request Number of VVS-Enabled Metaverse Service on Performances of Algorithms: Finally, we utilized the parameter setting **set 8** in Table IV, which increases the number of requests in the EVM from 50 to 250, to study its performance impact. In combination with Fig. 15(a) and (d), a clear trend is that the number of migration requests grows with the total number of requests, which leads to a gradual increase in the total cost. From Fig. 15(b), the running time of *spotlighter* is similar to that of *GOF* and *GOM*, which is much smaller than *ILPA*. The running time of *RC* is always the shortest among all algorithms in



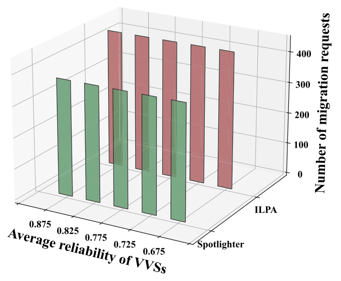
(a) The total cost



(b) The running time

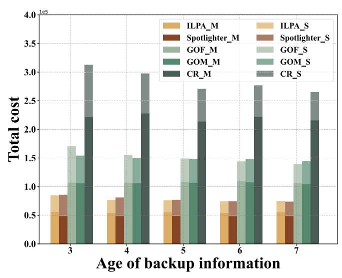


(c) The update frequency of VVSs

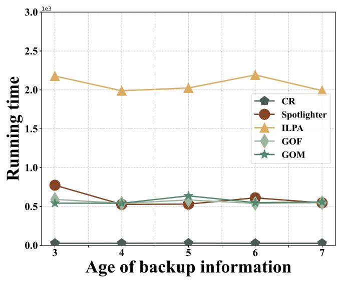


(d) The migration request number

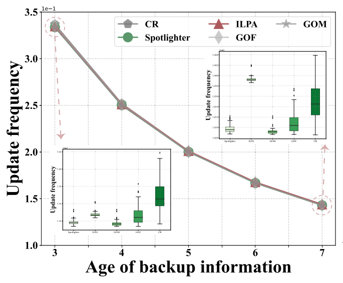
Fig. 10. Performance of different algorithms by varying the average reliability of VVSs from 0.675 to 0.875.



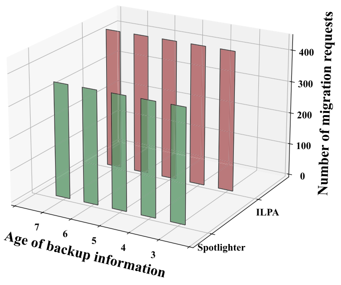
(a) The total cost



(b) The running time

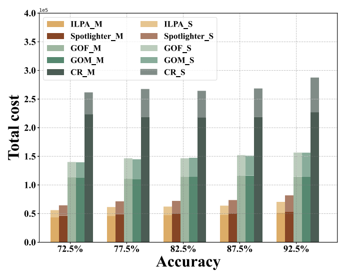


(c) The update frequency of VVSs

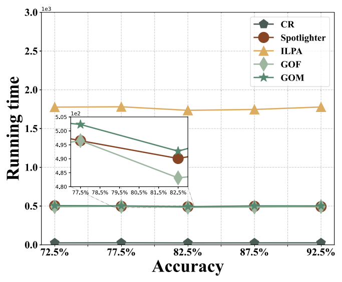


(d) The migration request number

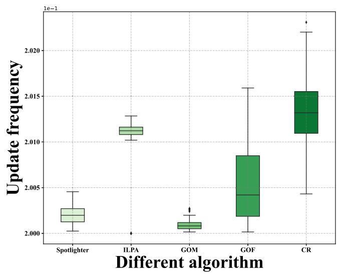
Fig. 11. Performance of different algorithms by varying the value of the AoBI from 3 to 7.



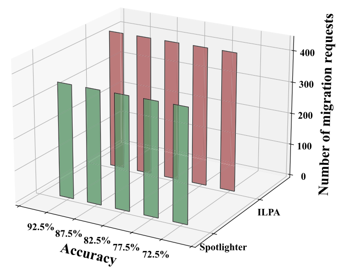
(a) The total cost



(b) The running time

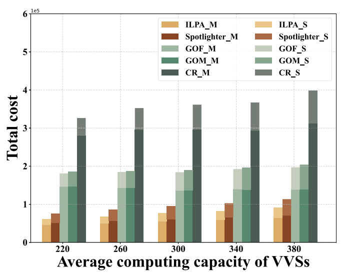


(c) The update frequency of VVSs

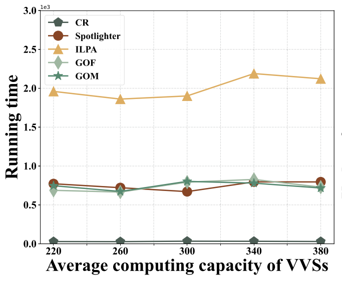


(d) The migration request number

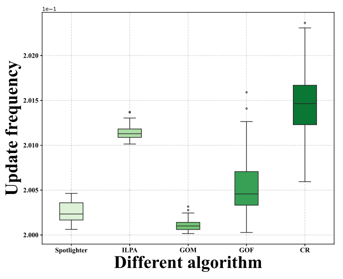
Fig. 12. Performance of different algorithms by varying accuracy from 75% to 95%.



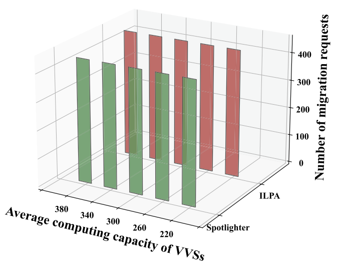
(a) The total cost



(b) The running time

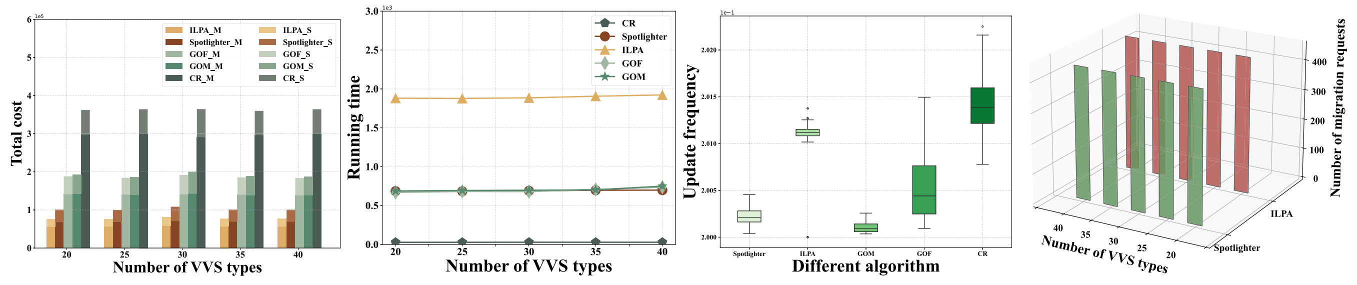


(c) The update frequency of VVSs



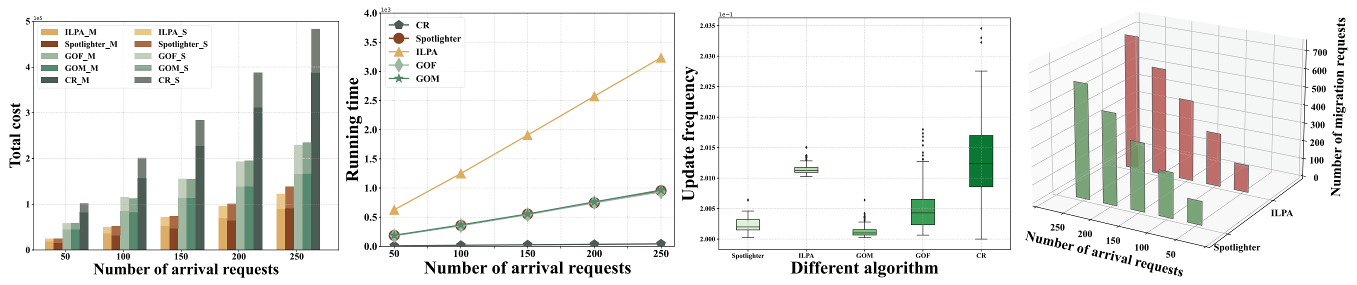
(d) The migration request number

Fig. 13. Performance of different algorithms by varying the average computing capacity of VVSs from 220 to 380.



(a) The total cost (b) The running time (c) The update frequency of VVSs (d) The migration request number

Fig. 14. Performance of different algorithms by varying the number of VVS types from 20 to 50.



(a) The total cost (b) The running time (c) The update frequency of VVSs (d) The migration request number

Fig. 15. Performance of different algorithms by varying the number of new arrival requests from 50 to 250.

all parameter settings. This is because the update frequencies and the migration locations of all backups are randomly generated without any strategy, so it has the shortest running time but the highest acceptance cost and update frequency. In Fig. 15(c), *spotlighter* has a slightly higher frequency than that of the *GOM* and less frequent than that of the *GOF*. This is because *GOM* always migrates backups to the nearest location to VVSs, which results in BVVSs always having the shortest communication latency, so there is no need to enhance update frequency to avoid AoBI conflicts. In contrast, deployment locations of all BVVSs remain fixed in the *GOF*, so the update frequency must be improved when a vehicle moves to a distant location.

VIII. Related Work

In this section, we survey the related studies and divide the works into two categories: vehicle metaverse, and age of information.

A. Vehicular Metaverse

Zhou et al. [27] recently outlined a macro yet blur framework of the vehicular metaverse combining individual vehicles and intelligent transportation systems, and introduced important enabler technologies. In addition, Li et al. [28] proposed an intelligent cockpit framework consisting of cognition, decision, and interaction layers, which can actively engage in empathic auditory regulation of driver anger and attention to improve driving safety. Ren et al. [29] designed an algorithm for improving sampling quality under different weather conditions to enable collective learning between heterogeneous vehicles by exploring quantum parallelism. Jiang et al. [30] proposed a game-based coded distributed computing framework that enables resource-limited devices to offload metaverse rendering tasks to other reliable devices. Chua et al. [31] considered a vehicular metaverse environment under which moving vehicles can download real-time virtual world updates from metaverse access points. They designed a reinforcement learning algorithm to minimize the total downloading time. Shi et al. [32] focused on both long-timescale and short-timescale decisions to strive for non-trivial trade-offing between service migrations and task rerouting of mobile devices during handovers, by adopting the Lyapunov optimization technique.

B. Age of Information

Yates et al. [23] summarized the latest optimization methods for updating the age of information (AoI) under various heterogeneous systems. Among them, Gindullina et al. [33] designed a two-source IoT monitoring system, in which a monitoring node collects the same data from both primary and alternate sources, they proposed an algorithm based on the Markov decision process to tradeoff the average freshness of information and energy cost. Li et al. [18] studied the AoI guaranteed scheduling problem in edge networks where source nodes are stationary. They developed a cyclic scheduler detection scheme for small scale networks and a polynomial mapping algorithm for large-scale networks to achieve the AoI guaranteed scheduling with a single base station and multiple sources. Li et al. [34]

recently considered the AoI-aware user satisfaction enhancement problem in a digital-twin empowered edge network, by developing performance-guaranteed approximation and online algorithms for user service request admissions under both static and dynamic request arrival settings. Abdel-Aziz et al. [35] proposed a probabilistic age of information model, investigated the tail distribution of the model, and designed a scheme based on Lyapunov optimization to minimize the transmission power while meeting the freshness demands of all vehicles. Moltafet et al. [11] formulated a mixed integer non-convex optimization problem that combines sampling frequency, transmit power, and sub-channel allocation, to minimize the transmit power within a finite time horizon while ensuring that the AoI of each sensor does not exceed a given threshold. Okegbile et al. [36] studied the human digital twin connectivity problem in the metaverse to optimize the synchronization age, training loss, privacy cost, and connectivity cost, by leveraging differential privacy, federated multi-task learning, and blockchain techniques.

IX. Conclusion

In this paper, we studied the long-term immersion VVS-enabled metaverse service provisioning problem, which is discretized into a series of single-slot optimization problems that then are formulated as integer linear programming. To ensure user immersion while facing various unknown failures, we designed a comprehensive resource scheduler called spotlighter. Specifically, we first developed a metaverse service home prediction algorithm trained on a real dataset with vehicular mobility traces to predict the real-time connected access points of each vehicle in the next time slot. We then devised a VVS migration algorithm with a provable approximation ratio by adopting the randomized rounding technique to simultaneously migrate VVSs requested by moving vehicles to minimize the VVS total migration cost consisting of both VVS re-instantiation cost and traffic transmission cost. We thirdly proposed a heuristic algorithm for AoBI-guaranteed to ensure AoBIs of all services, which selects a backup migration strategy or a backup update strategy adaptively to minimize the total backup synchronization cost. We finally conducted theoretical analysis and empirical simulation to evaluate the performance of the proposed algorithms, using real datasets. Simulation results show that the proposed algorithms are promising.

Acknowledgments

The authors appreciate the three anonymous referees and the associate editor for their constructive comments and invaluable suggestions, which help us to improve the quality and presentation of the paper greatly.

References

[1] M. Xu et al., "A full dive into realizing the edge-enabled metaverse: Visions, enabling technologies, and challenges," *IEEE Commun. Surveys Tut.*, vol. 25, no. 1, pp. 656–700, First Quarter 2023.

[2] F. Tian, X. Zhang, J. Liang, and Z. Yang, "Bidirectional service function chain embedding for interactive applications in mobile edge networks," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 3964–3980, May 2024.

[3] Y. Han et al., "A dynamic hierarchical framework for IoT-assisted digital twin synchronization in the metaverse," *IEEE Internet Things J.*, vol. 10, no. 1, pp. 268–284, Jan. 2023.

[4] S. D. Okegbile, J. Cai, and A. S. Alfa, "Practical byzantine fault tolerance-enhanced blockchain-enabled data sharing system: Latency and age of data package analysis," *IEEE Trans. Mobile Comput.*, vol. 23, no. 1, pp. 737–753, Jan. 2024.

[5] W. Zhang et al., "Deep reinforcement learning based resource management for DNN inference in industrial IoT," *IEEE Trans. Veh. Technol.*, vol. 70, no. 8, pp. 7605–7618, Aug. 2021.

[6] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark analysis of representative deep neural network architectures," *IEEE Access*, vol. 6, pp. 64270–64277, 2018.

[7] Y. Qiu, J. Liang, V. C. Leung, and M. Chen, "Online security-aware and reliability-guaranteed ai service chains provisioning in edge intelligence cloud," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 5933–5948, May 2024.

[8] Y. Qiu, J. Liang, V. C. M. Leung, X. Wu, and X. Deng, "Online reliability-enhanced virtual network services provisioning in fault-prone mobile edge cloud," *IEEE Trans. Wireless Commun.*, vol. 21, no. 9, pp. 7299–7313, Sep. 2022.

[9] C.-H. Tsai and C.-C. Wang, "Distribution-oblivious online algorithms for age-of-information penalty minimization," *IEEE/ACM Trans. Netw.*, vol. 31, no. 4, pp. 1779–1794, Aug. 2023.

[10] Y. Sun, E. Uysal-Biyikoglu, R. D. Yates, C. E. Koksal, and N. B. Shroff, "Update or wait: How to keep your data fresh," *IEEE Trans. Inf. Theory*, vol. 63, no. 11, pp. 7492–7508, Nov. 2017.

[11] M. Moltafet, M. Leinonen, M. Codreanu, and N. Pappas, "Power minimization for age of information constrained dynamic control in wireless sensor networks," *IEEE Trans. Commun.*, vol. 70, no. 1, pp. 419–432, Jan. 2022.

[12] Z. Xu, W. Liang, M. Huang, M. Jia, S. Guo, and A. Galis, "Efficient NFV-enabled multicasting in SDNs," *IEEE Trans. Commun.*, vol. 67, no. 3, pp. 2052–2070, Mar. 2019.

[13] A. Mahgoub, K. Shankar, S. Mitra, A. Klimovic, S. Chaterji, and S. Bagchi, "Sonic: Application-aware data passing for chained serverless applications," in *Proc. USENIX Annu. Tech. Conf.*, 2021, pp. 285–301.

[14] Z. Xu et al., "Stateful serverless application placement in MEC with function and state dependencies," *IEEE Trans. Comput.*, vol. 72, no. 9, pp. 2701–2716, Sep. 2023.

[15] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 21–25.

[16] D. Naboulsi and M. Fiore, "Characterizing the instantaneous connectivity of large-scale urban vehicular networks," *IEEE Trans. Mobile Comput.*, vol. 16, no. 05, pp. 1272–1286, May 2017.

[17] J. Li et al., "Budget-aware user satisfaction maximization on service provisioning in mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 12, pp. 7057–7069, Dec. 2023.

[18] C. Li et al., "Scheduling with age of information guarantee," *IEEE/ACM Trans. Netw.*, vol. 30, no. 5, pp. 2046–2059, Oct. 2022.

[19] Y. Sun, I. Kadota, R. Talak, and E. Modiano, "Age of information: A new metric for information freshness," *Synth. Lectures Commun. Netw.*, vol. 12, no. 2, pp. 1–224, 2019.

[20] China Internet Network Information Center, "The 45th China statistical report on internet development," 2020. Accessed: Jan. 7, 2021. [Online]. Avaialble: http://www.cac.gov.cn/2020-04/27/c_1589535470378587.htm

[21] M. Huang, W. Liang, Y. Ma, and S. Guo, "Maximizing throughput of delay-sensitive NFV-enabled request admissions via virtualized network function placement," *IEEE Trans. Cloud Comput.*, vol. 9, no. 4, pp. 1535–1548, Dec. 2021.

[22] E. Fountoulakis, N. Pappas, M. Codreanu, and A. Ephremides, "Optimal sampling cost in wireless networks with age of information constraints," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2020, pp. 918–923.

[23] R. D. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano, and S. Ulukus, "Age of information: An introduction and survey," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 5, pp. 1183–1210, May 2021.

[24] C. U. Manual, "Ibm ilog cplex optimization studio," *Version*, vol. 12, no. 1, pp. 1987–2018, 1987.

[25] A. Mpatziakas, A. Sinanis, I. Hamlatzis, A. Drosou, and D. Tzovaras, "AI-based mechanism for the predictive resource allocation of V2X related network services," in *Proc. 18th Int. Conf. Netw. Serv. Manage.*, 2022, pp. 282–288.

[26] M. Huang, W. Liang, X. Shen, Y. Ma, and H. Kan, “Reliability-aware virtualized network function services provisioning in mobile edge computing,” *IEEE Trans. Mobile Comput.*, vol. 19, no. 11, pp. 2699–2713, Nov. 2020.
[27] P. Zhou et al., “Vetaverse: A survey on the intersection of metaverse, vehicles, and transportation systems,” 2022, *arXiv:2210.15109*.
[28] W. Li et al., “Intelligent cockpit for intelligent vehicle in metaverse: A case study of empathetic auditory regulation of human emotion,” *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 53, no. 4, pp. 2173–2187, Apr. 2023.
[29] Y. Ren, R. Xie, F. R. Yu, T. Huang, and Y. Liu, “Quantum collective learning and many-to-many matching game in the metaverse for connected and autonomous vehicles,” *IEEE Trans. Veh. Technol.*, vol. 71, no. 11, pp. 12128–12139, Nov. 2022.
[30] Y. Jiang et al., “Reliable distributed computing for metaverse: A hierarchical game-theoretic approach,” *IEEE Trans. Veh. Technol.*, vol. 72, no. 1, pp. 1084–1100, Jan. 2023.
[31] T. J. Chua, W. Yu, and J. Zhao, “Resource allocation for mobile metaverse with the Internet of Vehicles over 6G wireless communications: A deep reinforcement learning approach,” in *Proc. IEEE 8th World Forum Internet Things*, 2022, pp. 1–7.
[32] Y. Shi, C. Yi, R. Wang, Q. Wu, B. Chen, and J. Cai, “Service migration or task rerouting: A two-timescale online resource optimization for MEC,” *IEEE Trans. Wireless Commun.*, vol. 23, no. 2, pp. 1503–1519, Feb. 2024.
[33] E. Gindullina, L. Badia, and D. Gündüz, “Average age-of-information with a backup information source,” in *Proc. IEEE 30th Annu. Int. Symp. Pers., Indoor Mobile Radio Commun.*, 2019, pp. 1–6.
[34] J. Li et al., “AOI-aware user service satisfaction enhancement in digital twin-empowered edge computing,” *IEEE/ACM Trans. Netw.*, vol. 32, no. 2, pp. 1677–1690, Apr. 2024.
[35] M. K. Abdel-Aziz, S. Samarakoon, C.-F. Liu, M. Bennis, and W. Saad, “Optimized age of information tail for ultra-reliable low-latency communications in vehicular networks,” *IEEE Trans. Commun.*, vol. 68, no. 3, pp. 1911–1924, Mar. 2020.
[36] S. D. Okegbile, J. Cai, H. Zheng, J. Chen, and C. Yi, “Differentially private federated multi-task learning framework for enhancing human-to-virtual connectivity in human digital twin,” *IEEE J. Sel. Areas Commun.*, vol. 41, no. 11, pp. 3533–3547, Nov. 2023.

Yu Qiu received the BSc degree from the Tianjin University of Technology, Tianjin, China, in 2020, the ME degree in computer technology from the Guangxi University, Nanning, China, in 2023. He is currently working toward the PhD degree in artificial intelligence with the South China University of Technology, Guangzhou, China. His research interests include metaverse, edge intelligence, network function virtualization, and optimization theory.

Min Chen (Fellow, IEEE) has been a full professor with School of Computer Science and Engineering, South China University of Technology. He is also the director of Embedded and Pervasive Computing (EPIC) Lab, Huazhong University of Science and Technology. He is the founding Chair of IEEE Computer Society Special Technical Communities on Big Data. He was an assistant professor in School of Computer Science and Engineering, Seoul National University before he joined HUST. He is the chair of IEEE Globecom 2022 eHealth Symposium. His Google Scholar Citations reached 40,000+ with an h-index of 96. His top paper was cited 4,304+ times. He was selected as Highly Cited Researcher from 2018 to 2022. He got IEEE Communications Society Fred W. Ellersick Prize, in 2017, the IEEE Jack Neubauer Memorial Award in 2019, and IEEE ComSoc APB Oustanding Paper Award in 2022. He is a Fellow IET.

Hebing Huang received the BE degree from Beijing Information Science and Technology University, Beijing, China, in 2021. He is currently working toward the MS degree in electronic and information engineering with Guangxi University, Nanning, China, since 2022. His research interests include mobile edge cloud, network function virtualization, and deep learning.

Weifa Liang (Senior Member, IEEE) received the BSc degree from Wuhan University, China, in 1984, the ME degree from the University of Science and Technology of China, in 1989, and the PhD degree from the Australian National University, in 1998, all in computer science. He is a full professor in the Department of Computer Science, City University of Hong Kong. Prior to that, he was a full professor in the Australian National University. His research interests include design and analysis of energy efficient routing protocols for wireless ad hoc and sensor networks, Mobile Edge Computing (MEC), Network Function Virtualization (NFV), Internet of Things and digital twins, design and analysis of parallel and distributed algorithms, approximation algorithms, and graph theory. He currently serves as an editor of *IEEE Transactions on Communications*.

Junbin Liang received the BE and MS degrees from Guangxi University, Nanning, China, and the PhD degree from Central South University, Changsha, China, in 2000, 2005, and 2010, respectively. He was a visiting professor with the University of British Columbia from 2019 to 2020. He is currently a professor with Guangxi University, Nanning, China. His research interests include sensor-cloud systems, fog computing, and distributed computing.

Yixue Hao (Member, IEEE) received the PhD degree in computer science from HUST, Wuhan, China, in 2017. He is an associate professor in the School of Computer Science and Technology, Huazhong University of Science and Technology (HUST). His Google Scholar Citations reached 5851+ with an h-index of 28 and i10-index of 42. He was named in Clarivate Analytics Highly Cited Researchers List, in 2020. His current research interests include cognitive computing, edge computing, and multi-agent reinforcement learning.

Dusit Niyato (Fellow, IEEE) received the BEng degree from the King Mongkuts Institute of Technology Ladkrabang (KMITL), Thailand, in 1999 and the PhD degree in electrical and computer engineering from the University of Manitoba, Canada, in 2008. He is a professor in the School of Computer Science and Engineering, Nanyang Technological University, Singapore. His research interests are in the areas of sustainability, edge intelligence, decentralized machine learning, and incentive mechanism design.