Dependency-Aware Service Migration for Backhaul-Free Vehicular Edge Computing Networks

Qibing Fan, Li Chen, *Senior Member, IEEE*, Changsheng You, *Member, IEEE*, Yunfei Chen, *Senior Member, IEEE*, and Huarui Yin, *Member, IEEE*

***Abstract*—Vehicular edge computing (VEC) is a promising paradigm to improve vehicular services through offloading complex computation tasks to the edge servers. However, the high mobility of vehicles requires frequent service migration among edge servers to guarantee uninterrupted services when vehicles traverse multiple cells. This brings great challenges. In this paper, we design a dependency-aware backhaul-free migration scheme to enable service migration without relying on backhaul with constraints on task dependencies. Specifically, the vehicle proactively fetches the migrated results based on task dependencies from the original server and migrates the results to its dynamically connected servers along the traveling path. Considering the incurred intermittent communication and computation due to vehicle mobility, a joint offloading and migration optimization problem for determining the time to offload tasks and fetch results is formulated with a time-varying Markov decision process (MDP) to minimize the total energy consumption. Time-varying transition probability functions are derived to characterize the dynamics during intermittent offloading and fetching. Based on the MDP framework, an efficient online value iteration algorithm is developed by exploiting temporal correlation to estimate the time-varying value functions. Simulation results demonstrate that our proposed algorithm can achieve superior energy-saving performance compared to the baseline online schemes.**

***Index Terms*—Backhaul-free network, Markov decision process, service migration, task dependency, vehicular edge computing.**

I. Introduction

WITH the rapid development of the Internet of Vehicles (IoV) and autonomous driving, various vehicular applications are emerging, such as image-aided navigation, intelligent vehicle control, and augmented vehicular reality [1]. These high-complexity applications require explosive computation resources and stringent time delays, yet it is commonly known that the computation capabilities of vehicles are limited and insufficient to meet such requirements. Vehicular edge computing (VEC) is proposed as a promising paradigm by deploying computing services in the close proximity to the vehicle to greatly reduce the latency [2]–[4].

The research of VEC can be traced back to the investigations related to mobile edge computing (MEC), a more general computing paradigm targeting not only vehicles but also other mobile devices. The state-of-the-art works on MEC from the communication perspective have been summarized in [5]. For example, for a single-user MEC system, an energy-optimal binary offloading strategy was proposed in [6], [7] by comparison between the energy consumption of offloading and local execution. To enable fine-grained computation offloading, bitwise independence in task partitioning was considered in [8], [9] and a resource allocation scheme was proposed for a multi-user MEC system. Liu et al. [10] proposed a new low-latency and reliable communication-computing system design for enabling mission-critical applications. The joint task offloading and resource allocation in the multi-user collaborative MEC network was investigated in [11] to minimize the total energy consumption under task delay constraints.

When the MEC technology is applied to the vehicular network, VEC arises. For the architecture design, the vehicle-assisted architecture that leverages vehicles as the infrastructures for communication and computation was first proposed in [12] to make better use of the underutilized resources of numerous vehicles. A cloud-assisted VEC architecture was developed in [3] to enable collaborative computation across cloud, edge, and vehicles. Based on various architectures, optimal control under VEC is considered. Considering the costs of both vehicles and edge servers, a dual-side offloading decision and resource allocation scheme was proposed in [13] for a single server system. To avoid performance degradation due to overload, the authors in [14] proposed a joint load balancing and offloading algorithm for maximizing system utility in a multi-server system. In [15], a task scheduling problem was investigated under a highly dynamic vehicular network to minimize the system energy consumption while satisfying task latency constraints. For the consideration of VEC realization, several enabling technologies have been introduced. Liu *et al.* [16] proposed a VEC network enabled by software-defined networking (SDN) to provide low-latency and high-reliability communication. Considering the requirements of various vehicle applications, network slicing was applied to VEC in [17] to support network service differentiation and diversification.

As vehicles may traverse different cells due to high mobility, a key topic in VEC-related research is service migration [18], [19]. Specifically, the ongoing computation services need to be migrated to the dynamically connected servers/BSs as the vehicle travels to guarantee uninterrupted services. In [20], an optimal migration policy was designed based on the

This research was supported by Industrial Technology Basic Project of MIIT (Grant No. TC220A04M). (*Corresponding author: Li Chen*)

Q. Fan, L. Chen and H. Yin are with CAS Key Laboratory of Wireless-Optical Communications, University of Science and Technology of China, Hefei 230027, China (e-mail: fanqibing@mail.ustc.edu.cn; chenli87@ustc.edu.cn; yhr@ustc.edu.cn).

Changsheng You is with the Department of Electronic and Electrical Engineering, Southern University of Science and Technology (SUSTech), Shenzhen 518055, China (e-mail: youcs@sustech.edu.cn).

Y. Chen is with the Department of Engineering, University of Durham, Durham DH1 3LE, U.K. (e-mail: Yunfei.Chen@durham.ac.uk).

Markov decision process (MDP) to achieve a good tradeoff between the migration cost and quality of service (QoS). The optimal policy was proved to be threshold-based in [21]. To proactively reshape the distribution of resource demands, mobility optimization was introduced into the service migration problem in [22]. Using both vehicle-to-vehicle and vehicle-to-infrastructure communications, Wang *et al.* [23] optimized task offloading and migration decisions based on game theory. As the service migration process involves BS handover, some studies focus on handover management. A vertical handover protocol based on Proxy Mobile IPv6 (PMIPv6) and IEEE 802.21 Media Independent Handover (MIH) standard is proposed in [24], which uses the received signal strength (RSS) and dynamic thresholds to trigger the handover. A blockchain-based handover authentication protocol for Vehicular ad-hoc networks (VANETs) is designed in [25] to guarantee the security of V2I communications.

Most of the above-mentioned works have made two assumptions. First, all the base stations are required to be deployed with backhaul. This will incur a significant infrastructure cost due to installation obstacles of fiber backhaul links [26], [27], especially in the dense VEC networks where base stations are densely deployed to increase the network capacity and provide ubiquitous services for vehicles. To avoid migration by backhaul, Zhang *et al.* [4] proposed to pre-offload tasks by multihop vehicle-to-vehicle connections, and the parked and moving vehicles were utilized for computation and communication in [12], [28]. Second, the IoV applications consist of independent subtasks [20], [29]. Nevertheless, in practice, inter-task dependencies exist in most IoV applications, where the outputs of some subtasks are the inputs of others. Taking the example of monitoring abnormal driving behavior in public safety [30], the vehicle first offloads various data collected by the on-board sensors, such as in-vehicle audio, video, and driving trajectory, to the edge for behavior feature extraction. The edge then performs a fusion analysis and sends alerts to the driver and passengers. Due to the continuity of the behavior monitoring application, the behavior analysis subtask relies on behavior data extracted by both the current and previous data processing subtasks, resulting in complex inter-task dependencies. However, traditional service migration methods generally only focus on whether to migrate [20], [29], ignoring the question of how much to migrate. This is essentially because they overlook inter-task dependencies, making it impossible to provide an explicit mathematical description of the migrated intermediate results. Consequently, achieving fine-grained service migration without relying on backhaul under the task dependency model is an urgent problem that deserves attention.

In this paper, we consider a multi-cell VEC network without backhaul and aim to jointly optimize computation offloading and service migration with inter-task dependency. The joint optimization addresses the following challenges. First, the results of multiple associated subtasks need to be migrated together to the same server due to task dependency. Second, due to the high mobility of vehicles, intermittent wireless connections may result in unsuccessful computation offloading and service migration. To enable service migration without backhaul but with task dependency, we propose an indirect migration scheme, where the vehicle proactively fetches intermediate results based on task dependencies from the original server and migrates the results to its dynamically connected servers/BSs along its traveling path. Accounting for the possible unsuccessful migrations of dependent results, some subtasks are recomputed after the BS handover, which leads to intermittent computation. To tackle intermittent communication and computation, the vehicle independently makes online decisions on when to offload tasks and fetch results based on the statistical information of the vehicular movement. The joint optimization problem minimizes the accumulated energy consumption under a highly dynamic environment of computation and communication.

The main contributions of this paper are summarized as follows.

- **Dependency-aware indirect migration scheme.** We first model the sequential offloading process of a successive task based on general task graphs. Based on this, we propose a novel dependency-aware indirect migration scheme to enable service migration without backhaul. Considering the effects of intermittent computation and communication, an indirect migration design on offloading and fetching is proposed to achieve efficient offloading and migration.
- **Joint optimization of computation offloading and results fetching based on MDP.** We study the joint offloading and migration problem to determine the time to offload tasks and fetch results to minimize the total energy consumption of completing all subtasks in a prior task graph. The optimization problem is formulated based on a time-varying MDP accounting for time-varying vehicle movement and wireless channels. In the MDP, we first map the vehicle location into the BS index to reduce state dimensions. Then, time-varying transition probability functions are derived to characterize the dynamics during intermittent offloading and fetching.
- **Energy-efficient online offloading and fetching policy.** The formulated energy minimization problem is solved using an MDP-based online value iteration (OVI) algorithm to obtain a real-time offloading and fetching policy, where the time-varying value function in time-varying MDP is estimated based on temporal correlation. In the proposed algorithm, an extra constraint to the value function is also used to avoid a selfish policy. Simulation results confirm that our proposed algorithm can achieve superior energy-saving performance compared to the oracle online schemes.

The remainder of this paper is organized as follows. The VEC system model is described in Section II. The proposed indirect migration scheme is depicted in detail in Section III. The MDP formulation and the corresponding algorithm for offloading and fetching decisions are discussed in Section IV. Simulation results are provided to evaluate the performance of the proposed algorithm in Section V. Finally, Section VI concludes the work.

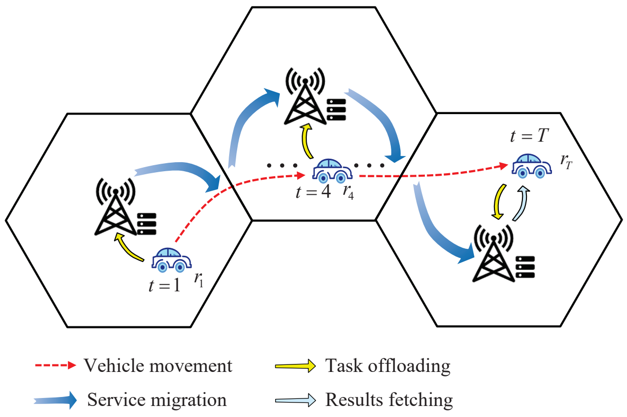


Fig. 1. System model.

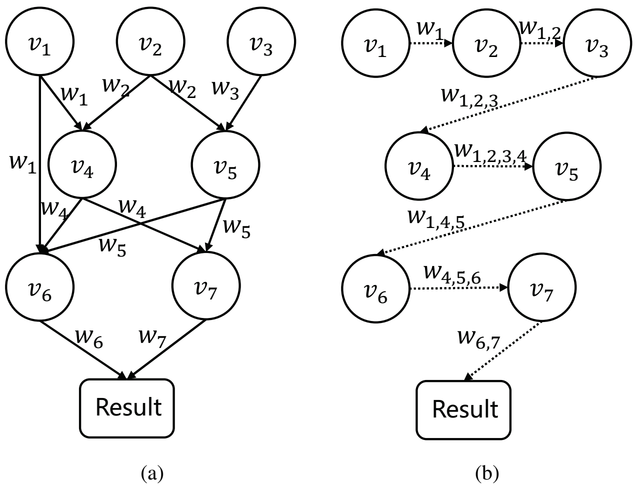


Fig. 2. An example task and the corresponding intermediate results for each subtask. (a) A task topology diagram describing the dependencies among subtasks. (b) The sequential offloading process graph with intermediate results marked on the dotted line. We abbreviate $\sum_{i=j,k,\ldots} w_i$ as $w_{j,k,\ldots}$ for brevity.

II. System Model

As depicted in Fig. 1, we consider a VEC system with a moving vehicle, N base stations (BSs), denoted by set $\mathcal{N} = \{1, 2, \ldots, N\}$. Each BS is equipped with a VEC server and can provide edge computing services. We assume that there are no overlapping coverage areas of BSs. Thus, the vehicle can only connect to one BS during the movement. The vehicle's computing task, such as analyzing images of the surrounding environment, needs to be offloaded to the BS due to the limited computing capability of the vehicle. The notations used in this paper are summarized in Table I.

The vehicle travels through multiple cells in the two-dimensional (2-D) region $\mathcal{A} \subset R^2$, as shown in Fig. 1. Time is discretized into multiple time slots, i.e., $t \in \mathcal{T} = \{1, 2, \cdots T\}$ and each time slot lasts T_0. The vehicle's location at slot t is denoted as $\mathbf{r}_t$. The mobility model can be described by a known probability distribution of the vehicle location at each time slot, denoted as $f_{\mathbf{R}_{(t)}}(\mathbf{r}_t)$.

We consider a sequential offloading process of a successive task. Specifically, it is assumed that the computing task can be further divided into M subtasks. They will be offloaded in sequence during the movement and are indexed as $[1, 2, \cdots, M]$ according to their offloading priority. The computation of a subtask requires the outputs of several previous subtasks, which is referred to as task dependencies. Considering the dependencies, we assume that the m-th offloaded subtask can be parameterized by $v_m \triangleq v(L_m, \alpha_m, \mathcal{I}_w(m))$, where L_m (in bits) denotes the size of subtask v_m, and α_m is computation intensity in terms of CPU cycles required to process a one-bit task computation, and $\mathcal{I}_w(m)$ is the intermediate results that get updated after the subtask v_m is completed. Theoretically, the output results of all completed tasks are included in the intermediate results. However, from the perspective of task dependencies, the intermediate results can be further simplified to reduce unnecessary storage. Specifically, if a completed task is not required by any subsequent task, it should be excluded from the intermediate results, otherwise, it should be retained. Therefore, the intermediate results $I_w(m)$ can be updated as

$$\mathcal{I}_w(m) = \sum_{i=1}^{m} w_i \cdot I\left(\text{succ}\left(v_i\right) \cap \{v_{m+1}, \ldots, v_M\} \neq \emptyset\right), \quad (1)$$

where w_i is the output size of each subtask v_i, and $\text{succ}(v_i)$ denotes the set containing all the successors depending on the output of v_i in the task topology diagram.

To illustrate the above sequential offloading process, we give an example in Fig. 2. Figure 2(a) shows the topology of the example task, where the directed arrow from node v_i to node v_j indicates that subtask v_j requires the output of subtask v_i, known as w_i. Figure 2(b) shows the sequential offloading process with corresponding intermediate results $\mathcal{I}_w(m)$ marked on each dotted line, where the sum of outputs $\sum_{i=j,k,\ldots} w_i$ is abbreviated as $w_{j,k,\ldots}$. After all the subtasks are offloaded and computed, the vehicle fetches the results of the last subtask, i.e., the target results of the computation task. The parameter information of all the subtasks is known a priori in our successive task offloading model. Unless otherwise specified, the tasks below refer to subtasks.

The mobility of the vehicle causes disruption to offloading. Specifically, the vehicle may not be able to offload all the tasks within the coverage area of a single BS considering the vehicle's mobility as well as the limited coverage of the BSs. In this case, after the vehicle moves out of the original BS coverage and connects to a new BS, the sequential offloading process will be interrupted due to task dependency limitations.

In order to guarantee continuous computation, a direct way is service migration [19], namely, migrating the intermediate results $\mathcal{I}_w(m)$ from the original BS to the vehicle's next BS along the traveling path.[1] Let $n_t \in \mathcal{N}$ denote the vehicle's associated BS at slot t. Clearly, results migration is triggered when $n_t \neq n_{t-1}$. The intermediate results are generally migrated via the backhaul network. However, providing backhaul connectivity for BSs incurs significant infrastructure costs in VEC networks. Therefore, we consider a backhaul-free scenario where each BS works independently without backhaul connectivity in this work. To support continuous computation

[1]The intermediate results are also referred to as running states in the literature related to service migration, including container or process migration.

TABLE I
DEFINITIONS OF NOTATIONS

Notation	**Definition**
N	The number of base stations
M	The number of subtasks in a task
T	The number of slots
T_0	The slot length
$\mathbf{z}_n$	The location of BS n
$\mathbf{r}_t$	The location of the vehicle at t-th slot
$d_n(t)$	The distance between vehicle and BS n at slot t
$f_c(n)$	The computing rate of the server n
n_t	The index of BS connected by the vehicle at the slot t
m_t	The index of task to be offloaded at the slot t
k_t	The index of the latest fetched task at slot t
L_m	The task size of task v_i
α_m	The computation intensity of task v_i
w_m	The output size of task v_i
$\mathcal{I}_w(m)$	The updated intermediate results after completing task m
B_u/B_d	The uplink/downlink bandwidth
P_u/P_r	The vehicle's transmit/receive power
P_B	The BS transmit power
N_0	The white Gaussian noise power
ψ_t	The small-scale fast fading power component
g_t	The log-normal shadowing coefficient
h_v/h_B	The antenna heights of the vehicle/BS
A/γ	The path-loss constant/exponent
$H_n(t)$	The channel power gain of the vehicle to BS n at slot t
$c_n^u(t)$	The uplink transmission rate in the BS n at slot t
$c_n^d(t)$	The downlink transmission rate in the BS n at slot t
$\mathbf{s}_t$	The composite state at slot t in time-varying MDP
a_t	The action at slot t in time-varying MDP
$P_t(\mathbf{s}'\|\mathbf{s},a)$	The state transition probability at slot t
D_t^m, D_t^u, D_t^w, D_t^d	The delay induced by migration, offloading, waiting, and fetching operation.
$E_t(\mathbf{s},a)$	The energy cost at slot t

without backhaul, we propose an indirect migration scheme that utilizes the connected vehicle to migrate intermediate results.

III. PROPOSED INDIRECT MIGRATION SCHEME

In this section, an indirect migration method is first proposed to achieve a more flexible migration without backhaul, where the vehicle proactively fetches intermediate results from the original BS and migrates them to the next BS as it travels. Considering the changing channels during the movement, the vehicle may experience channel distortion and suffer from time-out failure during the offloading and fetching process. To achieve efficient offloading and migration when intermittent computation and communication occur, an indirect migration design is proposed. After that, a detailed analysis of offloading and fetching decisions is provided for online decision-making.

A. Indirect Migration

The proposed indirect migration, as depicted in Fig. 3, is elaborated as follows. We denote $m_t \in \mathcal{M}$ as the index of the task to be offloaded at slot t. After the task v_{m_t} is offloaded and computed at a certain slot t, the vehicle proactively fetches intermediate results $\mathcal{I}_w(m_t)$ from the connected BS n_t via the downlink.[2] We refer to the operation of fetching $\mathcal{I}_w(m_t)$ as setting a checkpoint, and m_t is a checkpoint in successive tasks. As the vehicle connects to a new BS at slot t', it migrates the fetched results $\mathcal{I}_w(m_t)$ to the new BS via the uplink. Compared with the direct migration scheme, our proposed indirect migration can avoid the overhead of backhaul and is more flexible due to wireless connectivity.

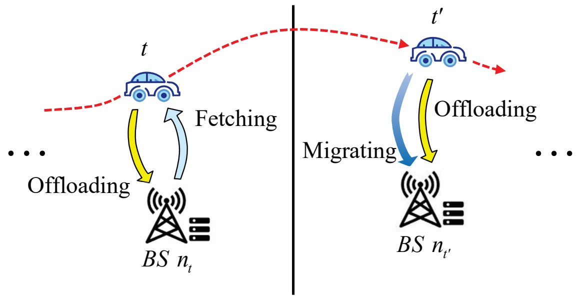


Fig. 3. The illustration of the indirect migration method.

However, there are still some challenges. First, some of the tasks may need to be recomputed after a handover. Specifically, when the vehicle migrates the intermediate results $\mathcal{I}_w(m_t)$ to the new BS at slot t', the BS can only start the computation from the next task following the checkpoint m_t, but not from the interrupted task indexed by $m_{t'}$. This is because the migrated intermediate results $\mathcal{I}_w(m_t)$ cannot satisfy the dependencies of tasks behind m_t+1. In this case, tasks indexed from m_t+1 to $m_{t'}$ need to be recomputed, which is termed as the move-out failure. In the highly dynamic scenario of vehicle mobility, the computation process of successive tasks is an intermittent computation process involving checkpoint settings, since the computation service is interrupted due to move-out and unfetched tasks need to be recomputed after a handover.

Second, the channel variation due to high vehicular mobility may lead to failures in offloading and fetching. In the proposed indirect migration, the results migration, task offloading, and results fetching are highly dependent on wireless channel conditions. Considering the varying path loss, shadowing, and multipath fading over time and space, the channel will change dynamically with vehicle movement. Therefore, the vehicle may experience severe channel distortion at some slots. We consider a bandwidth-limited scenario where the latency of task processing at a certain slot t may exceed slot length T_0, which is termed as the time-out failure. With the dynamic channel, the computation process of a successive task is also a process of intermittent communication. The channel power gain of the vehicle to BS n at slot t, denoted as $H_n(t)$, is assumed to be frequency-flat block fading, i.e., the channel remains static within each time slot but randomly varies over different slots.

[2]The intermediate results are fetched only for future migration because the vehicle itself doesn't require intermediate results but final results $\mathcal{I}_w(M)$.

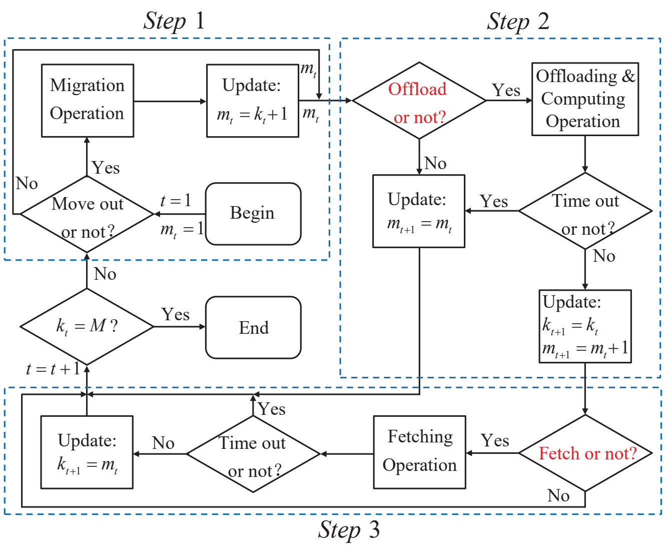


Fig. 4. The indirect migration design, including migration judgment, offloading decision, and fetching decision.

B. Indirect Migration Design

Considering the intermittent computation and communication, a key question is whether to offload and fetch at each slot. On one hand, in order to avoid an invalid offload resulting from a time-out failure, an offloading decision on whether to offload the task at each slot should be taken. On the other hand, after a task is offloaded and computed, the vehicle will make a fetching decision at each slot on whether to fetch the intermediate results for future migration to reduce the possibility of a move-out failure. We assume that the time-out failure is likely to occur during offloading and fetching phases, whereas the migration operation will not time out. It is reasonable because the migrated results are usually smaller than offloaded tasks in size and the migration operation is performed earlier compared with the fetching operation within a slot.

Based on the above intermittent offloading and fetching, we propose an indirect migration design to achieve efficient computation of M tasks under possible time-out failure and move-out failure, which is summarized in Fig. 4. To simplify, we denote the latest fetched task (checkpoint) up to slot t as k_t. It indicates the state of fetching and is updated after a successful fetch. For example, if the intermediate results $\mathcal{I}_w(m_t)$ are fetched at slot t without time-out, the k_{t+1} is updated to m_t, otherwise, it is not updated. At each slot t, there are three steps in our design:

1) **Step 1. Migration Judgement:** The vehicle system observes at the current slot t whether there is a BS handover. If the handover is observed, i.e., $n_{t-1} \neq n_t$, a migration of intermediate results $\mathcal{I}_w(k_t)$ is first performed at the new BS and thus the index of the task m_t is updated to $k_t + 1$.
2) **Step 2. Offloading Decision:** The system makes decisions at slot t whether to offload the task m_t. If it decides to offload m_t and it doesn't time out for the offloading and computing operations, the next task in the sequence will be offloaded at $t + 1$, i.e., $m_{t+1} = m_t + 1$.[3] On the contrary, if it times out or the system decides not to offload, the task needs to be recomputed at the next slot, i.e., $m_{t+1} = m_t$. Moreover, we keep $k_{t+1} = k_t$ until it is updated after a successful fetch.
3) **Step 3: Fetching Decision:** After a successful offload, the system decides whether to fetch the intermediate results $\mathcal{I}_w(m_t)$. If the system chooses to fetch the results and doesn't have a time-out failure in the fetching operation, the latest checkpoint k_{t+1} is updated to m_t. A failed fetch or a negative fetching decision will lead to a failure in updating k_{t+1}, that is, $k_{t+1} = k_t$.

After the results of the last task, $\mathcal{I}_w(M)$ is fetched at a certain slot t_0, i.e., $k_{t_0+1} = M$, the computation of M tasks is completed.

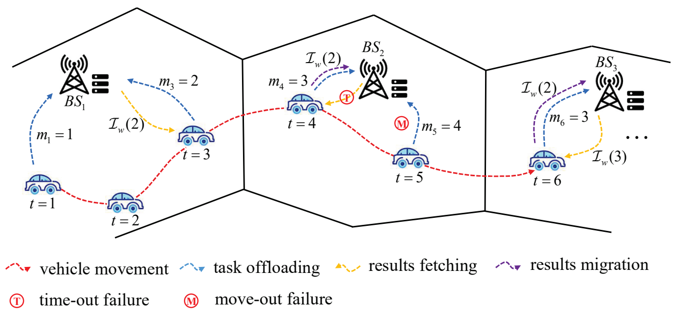


Fig. 5. An exemplary intermittent offloading and fetching process under the indirect migration design.

To illustrate the indirect migration design, an exemplary intermittent offloading and fetching process in the VEC network is given in Fig. 5.[4] First, located in the coverage of BS_1, the vehicle offloads task v_1 at slot $t = 1$ but offloads no task at slot $t = 2$ on account of the predicted time-out for offloading. After that, when the vehicle moves away from the BS_1 at slot $t = 3$, it offloads the task v_2 and fetches the intermediate results $\mathcal{I}_w(2)$, thus the latest fetched task at slot $t = 4$ is updated to $k_4 = 2$. Then, when it comes to $t = 4$, the vehicle moves out of the coverage of BS_1 and connects to BS_2. It migrates the latest fetched intermediate results $\mathcal{I}_w(2)$ and offloads task v_3 with the intention of fetching $\mathcal{I}_w(3)$, but ends up with a time-out failure during the fetching phase. At slot $t = 5$, the vehicle offloads the task v_4 without fetching $\mathcal{I}_w(4)$. As a result, a move-out failure occurs when the vehicle moves into the coverage of BS_3 at slot $t = 6$, and the tasks indexed after $k_6 = 2$, i.e., tasks v_3, v_4 are required to be recomputed. In other words, the process of task offloading rolls back to the offloading of task v_3 at slot $t = 6$.

C. Analysis of Offloading and Fetching Decisions

It is easy to see that the performance of the proposed scheme depends on the decisions on whether to offload and fetch. Furthermore, in order to adapt to the time-varying environment in real-time, we focus on online decision-making without

[3]For simplicity, we will regard the case of timing out during the computation operation as an offloading time-out.

[4]The trajectory in the example is hindsight.

requiring future information on user trajectory and wireless channel states in this work.

1) Offloading Decision Analysis: At first, it's difficult to determine whether to offload in real-time with limited channel information. Considering that no energy is consumed if a task is not offloaded, the vehicle can put off offloading to some future slots with higher transmission rates to achieve lower energy consumption. However, the global channel states are unknown to the vehicle in an online decision system.

In addition, the coupling between offloading and fetching decisions should also be considered. In particular, when the vehicle moves away from the BS, it is likely that it will suffer from an increasingly poor channel condition. Therefore, it becomes increasingly difficult for the vehicle to fetch intermediate results before moving out. If the computation of M tasks cannot be completed before the move-out, subsequent offloads are meaningless since the updated intermediate results cannot be fetched for migration.

2) Fetching Decision Analysis: For fetching decisions, the future movement trajectory is not available beforehand. Therefore, it is difficult to precisely locate the move-out slot t' and fetch the latest intermediate results $\mathcal{I}_w(t'-1)$ at one slot before to avoid recomputation. Even if a correct fetching decision is made, the fetching operation may be timed out when the vehicle suffers from a bad channel.

The optimization problem of offloading and fetching decisions is a sequential decision problem. Two commonly accepted powerful analytical tools for sequential decisions are Lyapunov optimization and Markov decision process (MDP). The Lyapunov method assumes a deterministic decision-making process, whereas in intermittent communication and computing systems caused by random vehicle movements, the decision-making process becomes stochastic. Fortunately, the latter one, MDP, is a promising approach to model sequential decision problems with probabilistically dynamic transitions in highly stochastic environments. Moreover, the Markov property in MDP that the result of an action does not depend on the previous actions and visited states, but only depends on the current state, can be satisfied in our problem after adjusting the state variables. Therefore, the MDP will be used to solve the problem.

IV. MDP-Based Offloading and Fetching Design

In this section, the problem of offloading and fetching will be formulated based on a discrete-time MDP model to optimize the performance of the proposed indirect migration scheme. First, we define the system states and actions. Time-varying state transition probabilities and immediate costs are then derived. Afterward, the joint optimization is formulated as a finite-horizon time-varying MDP problem to minimize the accumulated energy consumption and it is solved by employing an online value iteration algorithm.

A. State Space and Action Space

State space is not only the characterization of all possible system states but also the key to the curse of dimensionality for the MDP model. We map the vehicle's location into the corresponding BS, that is, transform an infinite-scale state space into a finite-scale state space, thereby leading to lower state dimensions [31]. Specifically, the state space is defined as $\mathbf{S}=\mathbf{\Phi}\times\mathcal{M}\times\mathcal{K}$, where $\mathbf{\Phi}=\{1,2,\ldots,N\}\times\{1,2,\ldots,N\}$ is the set of BSs connected by the vehicle at the latest two slots, and $\mathcal{M}=\{1,2,\ldots,M\}$ is the set of tasks to be offloaded and $\mathcal{K}=\{0,1,2,\ldots,M\}$ is the set of the latest fetched tasks in which the element 0 implies that no task has been fetched before t. [5] Correspondingly, the state can be characterized by a composite state $\mathbf{s}_t=(\mathbf{n}_t,m_t,k_t)\in\mathbf{S}$, where $\mathbf{n}_t,m_t,k_t$ denote the index of the BSs, the task to be offloaded and the latest fetched task at slot t, respectively. The sub-state $\mathbf{n}_t=(n_t,n_{t-1})$ denotes the indexes of BSs to which the vehicle is connected at slot t and $t-1$. Note that n_{t-1} is introduced into $\mathbf{s}_t$ to indicate a BS handover at slot t. In this case, the result of an action does not depend on the previous actions and visited states but on the current state, which satisfies the Markov property.

TABLE II
THE STATE AND ACTION FOR THE EXAMPLE IN FIG. 5 FROM SLOT 1 TO T

Slot: t	**State:** $\mathbf{s}_t=((n_t,n_{t-1}),m_t,k_t)$	**Action:** a_t
$t=1$	$\mathbf{s}_1=((1,1),1,0)$	$a_1=1$
$t=2$	$\mathbf{s}_2=((1,1),2,0)$	$a_2=0$
$t=3$	$\mathbf{s}_3=((1,1),2,0)$	$a_3=2$
$t=4$	$\mathbf{s}_4=((2,1),3,2)$	$a_4=2$
$t=5$	$\mathbf{s}_5=((2,2),4,2)$	$a_5=1$
$t=6$	$\mathbf{s}_6=((3,2),3,2)$	$a_6=2$

Then, the offloading and fetching decision at each slot t is expressed as action a_t in MDP. The action space $\mathbf{A}=\{0,1,2\}$ represents the set of possible actions. The system will decide to be on standby (i.e., $a_t=0$), or offloading without fetching (i.e., $a_t=1$), or performing both offloading and fetching (i.e., $a_t=2$) at slot t. Table II provides the detailed state and action information for the example in Fig. 5.

B. Time-Varying State Transition Probability

By applying action $a_t\in\mathbf{A}$ in state $\mathbf{s}_t\in\mathbf{S}$, the system makes a transition from $\mathbf{s}_t$ to a new state $\mathbf{s}_{t+1}\in\mathbf{S}$. Considering the highly dynamic environment with time-varying channels and mobility patterns, we characterize the dynamics using time-varying transition functions. The transition function at slot t is defined as $P_t:\mathbf{S}\times\mathbf{A}\times\mathbf{S}\rightarrow[0,1]$ and the state transition probability of ending up in state $\mathbf{s}_{t+1}$ after taking action a_t in state $\mathbf{s}_t$ at slot t is denoted as $P_t(\mathbf{s}_{t+1}\mid\mathbf{s}_t,a_t)$ or $P_t(\mathbf{s}'\mid\mathbf{s},a)$. It can be shown as

$$\begin{aligned}P_t(\mathbf{s}'\mid\mathbf{s},a)&=P_t(\mathbf{n}',m',k'\mid\mathbf{s},a)\\&=P_t(\mathbf{n}'\mid\mathbf{s},a)P_t(k'\mid\mathbf{n}',\mathbf{s},a)P_t(m'\mid\mathbf{n}',k',\mathbf{s},a)\,.\end{aligned}\tag{2}$$

where $P_t(\mathbf{s}'\mid\mathbf{s},a)$ is decomposed into three items, the BS transition probability, the offloading task transition probability,

[5]The state space of channel variations is not included in $\mathbf{S}$ because the channel is part of the external environment and its variation regularity is unknown to the vehicle. Furthermore, the continuity of the channel variations will also lead to an unacceptable infinite-scale state space.

and the latest fetched task transition probability in turn. Their detailed derivations are given below.

1) The BS Transition Probability: The BS transition probability $P_t(\mathbf{n}' \mid \mathbf{s}, a)$ can be simplified as $P_t(\mathbf{n}' \mid \mathbf{n}, a)$ because the BS state $\mathbf{n}'$ is independent of k and m. $\mathbf{n}'$ at slot $t+1$ is related to the location of the vehicle at t, so $P_t(\mathbf{n}' \mid \mathbf{n}, a)$ can be derived according to the Law of Total Probability as

$$P_t\left(\mathbf{n}' \mid \mathbf{n}, a\right) = \int_{\mathbf{r}_t \in \mathcal{A}_n} P_t\left(\mathbf{n}' \mid \mathbf{n}, a, \mathbf{r}_t\right) f_{\mathbf{R}_{(t)}}\left(\mathbf{r}_t \mid \mathbf{n}, a\right) d\mathbf{r}_t, \tag{3}$$

where $\mathbf{r}_t$ is the vehicle's location in the t-th slot, and $f_{\mathbf{R}_{(t)}}\left(\mathbf{r}_t \mid \mathbf{n}, a\right)$ is the conditional probability distribution function (pdf) of the vehicular location variable $\mathbf{R}_{(t)}$ given by the current BS state $\mathbf{n}$ and action a, and $\mathcal{A}_n$ denotes the coverage range of BS n_t. The location-based conditional state transition probability $P_t\left(\mathbf{n}' \mid \mathbf{n}, a, \mathbf{r}_t\right)$ can be derived as

$$\begin{aligned}
&P_t\left(\mathbf{n}' \mid \mathbf{n}, a, \mathbf{r}_t\right) \\
&\quad = P_t\left(\mathbf{n}'(1), \mathbf{n}'(2) \mid \mathbf{n}, a, \mathbf{r}_t\right) \\
&\quad = P_t\left(\mathbf{n}'(2) \mid \mathbf{n}, a, \mathbf{r}_t\right) \cdot P_t\left(\mathbf{n}'(1) \mid \mathbf{n}'(2), \mathbf{n}, a, \mathbf{r}_t\right) \\
&\quad = P_t\left(\mathbf{n}'(2) \mid \mathbf{n}(1)\right) \cdot P_t\left(\mathbf{n}'(1) \mid \mathbf{n}'(2), \mathbf{n}, a, \mathbf{r}_t\right) \\
&\quad = \begin{cases} P_t\left(n_{t+1} | n_t, a, \mathbf{r}_t\right), & \mathbf{n}'(2) = \mathbf{n}(1) \\ 0, & \text{otherwise,} \end{cases}
\end{aligned} \tag{4}$$

where $\mathbf{n}'(i), \mathbf{n}(i)$ denote the i-th term of $\mathbf{n}'$ and $\mathbf{n}$, and n_{t+1} is independent of n_{t-1} in the markov process. Further, $P_t\left(n_{t+1} \mid n_t, a, \mathbf{r}_t\right)$ is derived as

$$\begin{aligned}
P_t\left(n_{t+1} | n_t, a, \mathbf{r}_t\right) &= P_t(n_{t+1} | \mathbf{r}_t) \\
&= \int_{\mathbf{r}_{t+1} \in \mathcal{A}_{n_{t+1}}} f_{\mathbf{R}_{(t+1)} | \mathbf{R}_{(t)}}(\mathbf{r}_{t+1} | \mathbf{r}_t) d\mathbf{r}_{t+1} \\
&= \int_{\mathbf{r}_{t+1} \in \mathcal{A}_{n_{t+1}}} \frac{f_{\mathbf{R}_{(t+1)}, \mathbf{R}_{(t)}}(\mathbf{r}_{t+1}, \mathbf{r}_t)}{f_{\mathbf{R}_{(t)}}(\mathbf{r}_t)} d\mathbf{r}_{t+1},
\end{aligned} \tag{5}$$

where $f_{\mathbf{R}_{(t)}}(\mathbf{r}_t)$ is the pdf of vehicle's location in the t-th slot and $f_{\mathbf{R}_{(t+1)}, \mathbf{R}_{(t)}}(\mathbf{r}_{t+1}, \mathbf{r}_t)$ is the joint pdf of vehicle's locations in the t-th and $(t+1)$-th slot.

Further, we can use the Bayes theorem to derive the probability of the vehicle's location at slot t in (3) conditioned on the BS state and action as

$$\begin{aligned}
f_{\mathbf{R}_{(t)}}\left(\mathbf{r}_t \mid \mathbf{n}, a_t\right) &= f_{\mathbf{R}_{(t)}}\left(\mathbf{r}_t \mid n_t\right) \\
&= \frac{P_t\left(n_t \mid \mathbf{r}_t\right) \cdot f_{\mathbf{R}_{(t)}}\left(\mathbf{r}_t\right)}{P_t(n_t)} \\
&= \frac{P_t\left(n_t \mid \mathbf{r}_t\right) \cdot f_{\mathbf{R}_{(t)}}(\mathbf{r}_t)}{\int_{\mathbf{r}_t \in \mathcal{A}_{n_t}} f_{\mathbf{R}_{(t)}}(\mathbf{r}_t) d\mathbf{r}_t},
\end{aligned} \tag{6}$$

where $P_t\left(n_t | \mathbf{r}_t\right)$ indicates whether the location $\mathbf{r}_t$ belongs to the coverage range of the BS n_t. It returns the value of one if $\mathbf{r}_t$ is in the coverage range of BS n_t, otherwise zero, i.e.,

$$P_t\left(n_t \mid \mathbf{r}_t\right) = \begin{cases} 1, & \mathbf{r}_t \in \mathcal{A}_{n_t} \\ 0, & \mathbf{r}_t \notin \mathcal{A}_{n_t}. \end{cases} \tag{7}$$

Then, we substitute (4), (5), (6) and (7) into (3) to obtain the BS state transition probabilities $P_t(\mathbf{n}' \mid \mathbf{n}, a)$,

$$\begin{aligned}
&P_t(\mathbf{n}' \mid \mathbf{n}, a) \\
&\quad = \int_{\mathbf{r}_t \in \mathcal{A}_{n_t}} P_t\left(n_{t+1} \mid n_t, a, \mathbf{r}_t\right) f_{\mathbf{R}_{(t)}}\left(\mathbf{r}_t \mid n_t\right) d\mathbf{r}_t \\
&\qquad\qquad \cdot I(\mathbf{n}'(2) = \mathbf{n}(1)) \\
&\quad = \int_{\mathbf{r}_t \in \mathcal{A}_{n_t}} \left(\int_{\mathbf{r}_{t+1} \in \mathcal{A}_{n_{t+1}}} \frac{f_{\mathbf{R}_{(t+1)}, \mathbf{R}_{(t)}}\left(\mathbf{r}_{t+1}, \mathbf{r}_t\right)}{f_{\mathbf{R}_{(t)}}\left(\mathbf{r}_t\right)} d\mathbf{r}_{t+1} \right) \\
&\qquad \frac{P_t\left(n_t \mid \mathbf{r}_t\right) f_{\mathbf{R}_{(t)}}\left(\mathbf{r}_t\right)}{\int_{\mathbf{r}_t \in \mathcal{A}_{n_t}} f_{\mathbf{R}_{(t)}}\left(\mathbf{r}_t\right) d\mathbf{r}_t} d\mathbf{r}_t \cdot I(\mathbf{n}'(2) = \mathbf{n}(1)) \\
&\quad = \frac{\int_{\mathbf{r}_t \in \mathcal{A}_{n_t}} \int_{\mathbf{r}_{t+1} \in \mathcal{A}_{n_{t+1}}} f_{\mathbf{R}_{(t+1)}, \mathbf{R}_{(t)}}\left(\mathbf{r}_{t+1}, \mathbf{r}_t\right) d\mathbf{r}_{t+1} d\mathbf{r}_t}{\int_{\mathbf{r}_t \in \mathcal{A}_{n_t}} f_{\mathbf{R}_{(t)}}\left(\mathbf{r}_t\right) d\mathbf{r}_t} \\
&\qquad\qquad \cdot I(\mathbf{n}'(2) = \mathbf{n}(1)).
\end{aligned} \tag{8}$$

The BS state transition probability for any mobility model can be computed by (8), once $f_{\mathbf{R}_{(t)}}(\mathbf{r}_t)$ and $f_{\mathbf{R}_{(t)}, \mathbf{R}_{(t+1)}}(\mathbf{r}_t, \mathbf{r}_{t+1})$ are given.

2) The Latest Fetched Task Transition Probability: According to the proposed indirect migration design in Fig. 4, k' is updated only when a task is fetched without the time-out failure, otherwise, it will still equal to k. Therefore, the transition of k is independent of $\mathbf{n}'$ and the transition probability can be expressed as

$$\begin{aligned}
&P_t\left(k' \mid \mathbf{n}', \mathbf{s}, a\right) = P_t(k' \mid \mathbf{s}, a) \\
&\quad = \begin{cases} \left(1 - I_t^{\text{tout}}(\mathbf{s}, a)\right) I\left(a = 2\right), & k' = m \\ 1 - \left(1 - I_t^{\text{tout}}(\mathbf{s}, a)\right) I\left(a = 2\right), & k' = k \\ 0, & \text{otherwise,} \end{cases}
\end{aligned} \tag{9}$$

where $I(\cdot)$ returns the value of one if the condition in the bracket holds, and zero otherwise, and $I_t^{\text{tout}}(\mathbf{s}, a)$ indicates whether a fetching operation times out at slot t. Considering that the fetching operation is the last step in the edge computation process within a slot, $I_t^{\text{tout}}\left(\mathbf{s}, a\right)$ can be expressed as

$$I_t^{\text{tout}}(\mathbf{s}, a) = I\left(D_t(\mathbf{s}, a) > T_0\right), \tag{10}$$

where $D_t(\mathbf{s}, a)$ is the duration of a series of operation process at slot t including migration, computation offloading, computation waiting, and fetching operations, i.e.,

$$D_t\left(\mathbf{s}, a\right) = D_t^m\left(\mathbf{s}, a\right) + D_t^u\left(\mathbf{s}, a\right) + D_t^w\left(\mathbf{s}, a\right) + D_t^d\left(\mathbf{s}, a\right), \tag{11}$$

where $D_t^m\left(\mathbf{s}, a\right)$, $D_t^u\left(\mathbf{s}, a\right)$, $D_t^w\left(\mathbf{s}, a\right)$ and $D_t^d\left(\mathbf{s}, a\right)$ are the migration, offloading, waiting and fetching delays in the t-th slot, respectively. These delays are presented in Section IV-C.

3) The Offloading Task Transition Probability: In the indirect migration design, there are three possible values for the task m' which will be offloaded at slot $t+1$, namely the recomputed task $k'+1$, the next task $m+1$, and the original task m. Specifically, if there is a move-out at $t+1$, the task m' will be updated to $k'+1$. Conversely, if the vehicle stays in the original BS coverage area, m' is updated based on offloading decisions and time-out possibilities. On one hand, if the task m is offloaded and successfully computed, denoted as $I_t^{\text{succ}}(\mathbf{n}', \mathbf{s}, a) = 1$, m' is updated to be $m+1$, where

$$\begin{aligned} I_t^{\text{succ}}&(\mathbf{n}', \mathbf{s}, a) \\ &= I(a \neq 0) \cdot I\left(D_t^m(\mathbf{s}, a) + D_t^u(\mathbf{s}, a) + D_t^w(\mathbf{s}, a) \leq T_0\right). \end{aligned} \tag{12}$$

On the other hand, if the task m is not offloaded or the offloading operation is timed out, i.e., $I_t^{\text{succ}}(\mathbf{n}', \mathbf{s}, a) = 0$, the task m will be recomputed at slot $t+1$, i.e., $m' = m$. Notice that when offloading the final task M without suffering from the move-out failure, the vehicle will keep offloading it until the termination condition $k' = M$ is met, namely $m' = m = M$. Given the above-mentioned considerations, the offloading task transition probability $P_t(m' \mid \mathbf{n}', k', \mathbf{s}, a)$ can be given by (13), where I_{t+1}^{mout} indicates whether the vehicle moves out of the coverage of the original BS at $t+1$, namely

$$I_{t+1}^{\text{mout}}(\mathbf{n}') = 1 - I(n_{t+1} = n_t). \tag{14}$$

Note that an additional restriction, $m' \neq k'+1$, is added to the first two conditions in (13) to avoid overlapping conditions when k' is equal to m or $m-1$ in the third condition. Furthermore, it can be seen from (13) that the next state m_{t+1} is related to the current migration delay $D_t^m(\mathbf{s}, a)$. The current migration delay is incurred when the vehicle moves out of the coverage area of the original BS n_{t-1}, i.e., $n_t \neq n_{t-1}$. Therefore, the next state m_{t+1} depends on the past BS index n_{t-1}. To satisfy the required Markov property in MDP, we unbind m_{t+1} from the past state $\mathbf{s}_t$ by introducing the n_{t-1} into the state $\mathbf{s}_t$, as described in Section IV-A.

C. Time-Varying Immediate Cost

The vehicle needs to consume energy for data transmission after taking some action in a state. The cost function at slot t is defined as $E_t : \mathbf{S} \times \mathbf{A} \rightarrow \mathbb{R}$ which specifies the immediate communication energy cost for action a_t in state $\mathbf{s}_t$, abbreviated as $E_t(\mathbf{s}, a)$. The uplink transmission rate of the vehicle to BS n at slot t is given by [32]

$$\begin{aligned} c_n^u(t) &= B_u \log_2\left(1 + \frac{P_u H_n(t)}{N_0}\right), \\ c_n^d(t) &= B_d \log_2\left(1 + \frac{P_B H_n(t)}{N_0}\right), \end{aligned} \tag{15}$$

where B_u, B_d represent the uplink and downlink bandwidths and P_u, P_B denote the transmit power for the vehicle and BS, respectively, and N_0 is the noise power. $H_n(t)$ is the channel power gain between vehicle and BS n as [33]

$$H_n(t) = \psi_t g_t A d_n(t)^{-\gamma} \tag{16}$$

where ψ_t is the small-scale fast fading power component, assumed to be exponentially distributed with unit mean [34], g_t is a log-normal shadowing with a standard deviation ξ, A is the path loss constant, $d_n(t)$ is the distance between the vehicle and BS n at slot t, and γ is the decay exponent. The distance $d_n(t)$ can be calculated as

$$d_n(t) = \sqrt{\left\|\mathbf{r}_t - \mathbf{z}_n\right\|^2 + (h_v - h_B)^2}, \tag{17}$$

where $\mathbf{r}_t$ is the location of the vehicle at slot t, $\mathbf{z}_n$ is the location of the connected BS n, and h_v, h_B represent the antenna heights of the vehicle and BS n, respectively. It is worth noting that the location $\mathbf{r}_t$ of the vehicle is updated in real-time. In addition, we assume that the vehicle can acquire the uplink and downlink transmission rates $c_n^u(t)$ and $c_n^d(t)$ in real-time before taking an action in the state $\mathbf{s}_t$. Based on the above communication model, the delay of migration, offloading, computation waiting, and fetching operations can be given as follows.

First, the migration delay is incurred by the uplink transmission of the intermediate results $\mathcal{I}_w(k)$ if there is a move-out, i.e.,

$$D_t^m(\mathbf{s}, a) = \frac{I_t^{\text{mout}}(\mathbf{n}) \cdot \mathcal{I}_w(k)}{c_n^u(t)}. \tag{18}$$

The offloading delay, induced by the uplink transmission of the task m, can be expressed as

$$D_t^u(\mathbf{s}, a) = \frac{L_m \cdot (1 - I(a=0))}{c_n^u(t)}. \tag{19}$$

After that, the waiting delay is incurred by waiting for the task computation, which can be denoted by

$$D_t^w(\mathbf{s}, a) = \frac{L_m \cdot \alpha_m}{f_c(n)} \cdot (1 - I(a=0)), \tag{20}$$

where α_m is the required computation intensity of task m and $f_c(n)$ is the computing rate of the server n, which represents the available CPU cycle frequency of the VEC server for task processing. Finally, the fetching delay in the t-th slot is

$$D_t^d(\mathbf{s}, a) = \frac{\mathcal{I}_w(m) \cdot I(a=2)}{c_n^d(t)}. \tag{21}$$

Given the above four operation delays, the immediate energy cost under all the possible time-out cases can be given as

$$\begin{aligned} E_t^c = &I(\text{flag}=1)(P_u T_0) \\ &+ I(\text{flag}=2)\left(P_u\left(D_t^m + D_t^u\right)\right) \\ &+ I(\text{flag}=3)\left(P_u\left(D_t^m + D_t^u\right) + P_r\left(T_0 - D_t^m - D_t^u - D_t^w\right)\right) \\ &+ I(\text{flag}=4)\left(P_u\left(D_t^m + D_t^u\right) + P_r D_t^d\right) \end{aligned} \tag{22}$$

$$\begin{aligned} &P_t(m' \mid \mathbf{n}', k', \mathbf{s}, a) \\ &= \begin{cases} \left(1 - I_{t+1}^{\text{mout}}\right) \cdot I_t^{\text{succ}}, & m' = m+1 \leq M, m' \neq k'+1 \\ \left(1 - I_{t+1}^{\text{mout}}\right) \cdot \left(1 - I_t^{\text{succ}} + I(m = M)\right), & m' = m \leq M, m' \neq k'+1 \\ I_{t+1}^{\text{mout}} + \left(1 - I_{t+1}^{\text{mout}}\right) \cdot \left(I_t^{\text{succ}} \cdot I(k' = m) + \left(1 - I_t^{\text{succ}}\right) \cdot I(k' = m-1)\right), & m' = k'+1 \leq M \\ 0, & \text{otherwise,} \end{cases} \end{aligned} \tag{13}$$

where P_u and P_r are the vehicle's transmit and receive power, respectively, and the indicator function flag $(\mathbf{s}, a)$, which indicates different cases of time-out failure under $(\mathbf{s}, a)$, is defined as

$$\begin{aligned}&\text{flag}\left(\mathbf{s}, a\right)\\ &=\begin{cases}1 & D_t^m+D_t^u \geq T_0\\ 2 & D_t^m+D_t^u < T_0, D_t^m+D_t^u+D_t^w \geq T_0\\ 3 & D_t^m+D_t^u+D_t^w < T_0, D_t^m+D_t^u+D_t^w+D_t^d \geq T_0\\ 4 & D_t^m+D_t^u+D_t^w+D_t^d < T_0.\end{cases}\end{aligned} \tag{23}$$

From the perspective of uplink and downlink transmissions, (22) can be written as follows,

$$\begin{aligned}E_t\left(\mathbf{s}, a\right) =& P_u \cdot \min\left(D_t^m + D_t^u, T_0\right) + \\ & P_r \cdot \min\left(D_t^d, \max\left(T_0 - D_t^m - D_t^u - D_t^w, 0\right)\right).\end{aligned} \tag{24}$$

It is worth noting that the above delays under all the state-action pairs $(\mathbf{s}, a)$ can be computed based on known task parameters and real-time acquired transmission rate before a specific action is taken. In this way, the immediate cost function E_t as well as the transition function P_t is also available before the system takes an action based on (2), (8), (9), (13) and (24). Notice that in (8), the BS state transition function is computed based on the prior statistical knowledge of vehicle mobility.

D. Online MDP-Based Offloading and Fetching Algorithm

Based on the MDP model, we formulate an MDP optimization problem to minimize the expected accumulated communication energy consumption during T slots under the state transition function, which is given by

$$\min_{\pi} \mathcal{J}^{\pi} = \mathbb{E}\left(\sum_{t=1}^{T} E_t'\left(\mathbf{s}, \pi_t(\mathbf{s})\right)\right) \tag{25}$$

where π_t is a time-related deterministic policy, defined as $\pi_t : \mathbf{S} \to \mathbf{A}$, that is, $a_t = \pi_t(\mathbf{s}_t) \in \mathbf{A}$, and $\pi = \{\pi_t\}_{t\in\mathcal{T}}$, and the expectation $\mathbb{E}(\cdot)$ is taken to average all possible accumulated energy consumption in the random process, and $E_t'(\mathbf{s}, a)$ is defined as

$$E_t'\left(\mathbf{s}, a\right) = \begin{cases}E_t\left(\mathbf{s}, a\right), & \mathbf{s} \in \mathbf{S}_{\text{n-end}}\\ 0, & \mathbf{s} \in \mathbf{S}_{\text{end}},\end{cases} \tag{26}$$

where $\mathbf{S}_{\text{end}}$ is the set of ending states with $\mathbf{S}_{\text{end}} = \{\mathbf{s} \in \mathbf{S} \mid \mathbf{s}(4) = M\}$ and $\mathbf{S}_{\text{n-end}}$ is the set of non-ending states with $\mathbf{S}_{\text{n-end}} = \{\mathbf{s} \in \mathbf{S} \mid \mathbf{s}(4) < M\}$. Notice that we replace $E_t(\mathbf{s}, a)$ with $E_t'(\mathbf{s}, a)$ in (25) because no more energy will be consumed once the final task is fetched. Correspondingly, the ending state will not be transferred, i.e.,

$$P_t'(\mathbf{s}' \mid \mathbf{s}, a) = \begin{cases}P_t(\mathbf{s}' \mid \mathbf{s}, a), & \mathbf{s} \in \mathbf{S}_{\text{n-end}}\\ 0, & \mathbf{s} \in \mathbf{S}_{\text{end}}.\end{cases} \tag{27}$$

Due to the time-varying transition probabilities and costs as well as finite slots, the problem in (25) is a finite-horizon time-varying MDP problem. The MDP problem follows the following procedures. At each slot $t = 1, 2, \ldots, T$,

Algorithm 1 Online Value Iteration (OVI)

```
Input: Initial values $V_0(\mathbf{s}), \forall \mathbf{s} \in \mathbf{S}$. Initial time: $t = 1$. The
    number of iterations $K$.
Output: Generated policy $\pi_t, \forall t \in \mathcal{T}$.
 1: while $(t \leq T)$ do
 2:     Step 1: update value $V_t(\mathbf{s})$ by $K$ iterations of value
        iteration
 3:     Set $V_t^0(\mathbf{s}) = V_{t-1}(\mathbf{s}), \mathbf{s} \in \mathbf{S}$.
 4:     for $k = 1 : K$ do
 5:         Update $V_t^k(\mathbf{s}), \mathbf{s} \in \mathbf{S}$ as follows:
            $V_t^k(\mathbf{s}) = \min_{a\in\mathbf{A}} \left\{E_t'(\mathbf{s}, a) + \sum_{\mathbf{s}'\in\mathbf{S}} P_t'(\mathbf{s}' \mid \mathbf{s}, a)\, V_t^{k-1}(\mathbf{s}')\right\},$
 6:     end for
 7:     Set $V_t(\mathbf{s}) = V_t^K(\mathbf{s}), \mathbf{s} \in \mathbf{S}$.
 8:     Step 2: generate the policy $\pi_t$ at the current slot $t$ to
        minimize the estimated Q function
        $\pi_t(s) = \arg\min_{a\in\mathbf{A}} Q_t(\mathbf{s}, a)$
        $= \arg\min_{a\in\mathbf{A}} \left\{E_t'(\mathbf{s}, a) + \sum_{\mathbf{s}'\in\mathbf{S}} P_t'(\mathbf{s}' \mid \mathbf{s}, a)\, V_t(\mathbf{s}')\right\}.$
 9:     Set $t = t + 1$.
10: end while
```

1) the system observes the current state $\mathbf{s}$ and computes the cost function E_t' and the transition probability function P_t',
2) the system takes an action $a = \pi_t(\mathbf{s})$ based on the policy π_t and takes a cost $E_t'(\mathbf{s}, a)$,
3) the new state $\mathbf{s}' \in \mathbf{S}$ is drawn based on the transition probability distribution $P_t'(\cdot|\mathbf{s}, a)$.

The optimal policy at each slot t, denoted as π_t^*, can be determined by the Bellman optimality equation [35] as

$$V_t(\mathbf{s}) = \min_{a\in\mathbf{A}} \left\{E_t'(\mathbf{s}, a) + \sum_{\mathbf{s}'\in\mathbf{S}} P_t'\left(\mathbf{s}' \mid \mathbf{s}, a\right) V_{t+1}\left(\mathbf{s}'\right)\right\}, \tag{28}$$

where the optimal value $V_t(\mathbf{s})$ indicates the expected accumulated cost when starting in $\mathbf{s}$ at slot t and taking optimal actions thereafter. The optimal policy, given the current state $\mathbf{s}$, is given by

$$\pi_t^*(\mathbf{s}) = \arg\min_{a\in\mathbf{A}} \left\{E_t'(\mathbf{s}, a) + \sum_{\mathbf{s}'\in\mathbf{S}} P_t'\left(\mathbf{s}' \mid \mathbf{s}, a\right) V_{t+1}\left(\mathbf{s}'\right)\right\}. \tag{29}$$

Unfortunately, $V_{t+1}(\mathbf{s})$ is unavailable at the current slot t for any online system. To address the problem, we propose an Online Value Iteration (OVI) algorithm, which is inspired by the work in [36]. The main idea is to leverage the time-adjacent value $V_t(\mathbf{s})$ to approximate the value $V_{t+1}(\mathbf{s})$ considering the similarity of the environment at adjacent slots. In this way, (28) can be rewritten as

$$V_t(\mathbf{s}) \approx \min_{a\in\mathbf{A}} \left\{E_t'(\mathbf{s}, a) + \sum_{\mathbf{s}'\in\mathbf{S}} P_t'\left(\mathbf{s}' \mid \mathbf{s}, a\right) V_t\left(\mathbf{s}'\right)\right\}, \tag{30}$$

which can be solved by employing the well-known value iteration algorithm [35]. As illustrated in Algorithm 1, there are two steps at each slot t in the OVI algorithm.

1) **Step 1:** The algorithm runs K iterations of value iteration based on the immediate cost function E'_t, the transition probability function P'_t, and the previously estimated value function V_{t-1}, and generates a new estimate of the value function V_t.
2) **Step 2:** The algorithm generates an online policy on the basis of the current estimate of value function V_t and the immediate cost function E'_t and transition probability function P'_t.

Because there is no energy consumption if no task is offloaded, the algorithm may tend not to offload any task at each slot t (i.e., $\pi_t(\mathbf{s}) = 0$), referred to as a selfish offloading policy. To avoid this, we set the initial value $V_0(\mathbf{s})$ as

$$V_0(\mathbf{s}) = \begin{cases} 1, & \mathbf{s} \in \mathbf{S}_{\text{end}}, \\ \text{H}, & \mathbf{s} \in \mathbf{S}_{\text{n-end}}. \end{cases} \tag{31}$$

where H is a constant satisfying $\text{H} \gg \mathcal{J}^{\pi}$. In this case, the Q-value function $Q_1(\mathbf{s}, a)$ under the selfish offloading policy will always be as large as H since the initial state cannot be transferred to the ending states. Likewise, the subsequent $Q_t(\mathbf{s}, a)$ from $t = 2$ will be indirectly influenced by $V_0(\mathbf{s})$. Using this, the algorithm tends to choose a policy that can eventually guide the state to an ending state to avoid excessive costs, thus avoiding selfish offloading. Notice that the algorithm will dynamically adjust the offloading and fetching policy to adapt to the varying environments. For example, when the vehicle speed becomes larger or the channel becomes worse, a fetch decision ($a_t = 2$) will be preferred.

It can be proven that the proposed OVI algorithm converges, and the proof is presented in Appendix 1. Based on the pseudo-code presented in Algorithm 1, the computational complexity for each slot t is determined by the product of the complexity required to update the value function and the number of iterations. For an MDP problem with $|\mathbf{S}|$ states and $|\mathbf{A}|$ actions, the computational complexity required to update the value function $V_t^k(\mathbf{s})$ for each state $\mathbf{s} \in \mathbf{S}$ is $O(|\mathbf{S}| \cdot |\mathbf{A}|)$. Therefore, the total computational complexity of the OVI algorithm for each slot t is $O\left(|\mathbf{S}|^2 \cdot |\mathbf{A}| \cdot K\right)$. Through mapping the vehicle's locations into the corresponding BS under the prior mobility information, we transform an infinite-scale state space into a finite-scale state space, which contributes smaller $|\mathbf{S}|$ and lower algorithm complexity.

V. Numerical Results and Discussion

This section provides some simulation results to illustrate the performance of the proposed OVI scheme. We consider a highway scenario with a length of L, where $N = 5$ BSs are placed evenly on the roadside, as depicted in Fig. 6. The vehicle moves according to the ordered uniform distribution model [37], that is, the probability distribution of the vehicle's

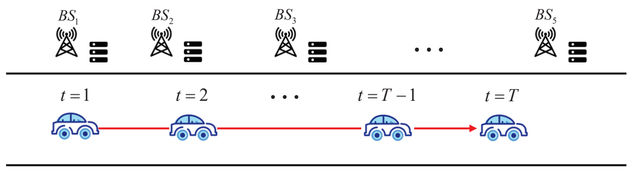


Fig. 6. An illustration of highway scenario.

location variable in the t-th timeslot during T_m movement timeslots is given by

$$f_{R_{(t)}}(r_t) = \frac{T_m!}{(t-1)!(T_m-t)!} \left[F_R(r_t)\right]^{t-1} \left[1 - F_R(r_t)\right]^{T_m - t} f_R(r_t), \tag{32}$$

where $f_R(r_t)$ and $F_R(r_t)$ are the probability density function (pdf) and the cumulative distribution function (cdf) under the uniform distribution of the vehicle location. Furthermore, the joint pdf of the location variable at slot t and slot $t+1$ can be derived based on the statistics knowledge on order statistics [38] :

$$f_{R_{(t)},R_{(t+1)}}(r_t, r_{t+1}) = \frac{T_m!}{(t-1)!(T_m-t-1)!} \left[F_R(r_t)\right]^{t-1} \cdot \left[1 - F_R(r_{t+1})\right]^{T_m-t-1} f_R(r_t) f_R(r_{t+1}). \tag{33}$$

In our simulation, the vehicle's movement trajectory is randomly generated 1000 times according to its mobility model in (32). The segment length and the movement timeslot length are set to be $L = 2000$ m and $T_m = 40$, respectively. The number of timeslots T equals to the number of vehicle movement timeslots T_m. The slot length is $T_0 = 2$ s and the average speed of the vehicle equals to $v = \frac{L}{T_m \times T_0} = 90$ km/h. For the task parameters, the number of divided subtasks is set to be $M = 7$. The subtask size is $L_m \in [8, 12]$ Mbits and the computation intensity is $\alpha_m \in [100, 1000]$ cycles/bit. The output size of each subtask is $w_m \in [1, 1.5]$ Mbits. The task topology is generated by reference [39]. Given the task topology, the offloading priority in the sequential offloading scenario can be determined by a topological sorting algorithm [39], and the corresponding intermediate results $\mathcal{I}_w(m)$ is generated according to (1).

We follow the simulation setup on the channel for the highway case detailed in 3GPP TR 36.885 [40]. The vehicle's transmit power is $P_u = 23$ dBm [33] and the receive power is $P_r = 26$ dBm [41]. Besides, the uplink and downlink channel bandwidth are $B_u = B_d = 20$ MHz [31] and the noise power is $N_0 = -114$ dBm [33]. In terms of the edge, the BS transmit power is $P_B = 30$ dBm [33] and the computing rate of each VEC server is $f_c = 10$ GHz [31]. On each generated movement trajectory, the transmission rate at each timeslot is randomly generated according to the communication model. Notice that the vehicle movement causes Doppler spread and random signal variations, posing a challenge to vehicular communication. Nevertheless, since the 3GPP TR 36.885 protocol has already used enhanced Demodulation Reference Signal (DMRS) technology to deal with high Doppler effects in vehicle communications, we assume

TABLE III
SIMULATION PARAMETERS

Parameter	Value
Highway segment length, L	2000 m
Movement timeslot length, T_m	40 slots
Simulation timeslot length, T	40 slots
Slot length, T_0	2 s
Number of BSs, N	5
Size of the subtask, L_m	$[8, 12]$ Mbits
Computation intensity of the subtask, α_m	$[100, 1000]$ cycles/bit
Output size of the subtask, w_m	$[0.8, 1.2]$ Mbits
Number of subtasks, M	7
Uplink/Downlink bandwidth, B_u/B_d	20 MHz
Vehicle antenna height, h_v	1.5 m
Vehicle antenna gain	3 dBi
Vehicle receiver noise figure	9 dB
BS antenna height, h_B	25 m
BS antenna gain	8 dBi
BS receiver noise figure	5 dB
Pathloss model	$128.1 + 37.6 \log_{10} d$, d in km
Shadowing distribution	Log-normal
Shadowing standard deviation, ξ	8 dB
Fast fading	Rayleigh fading
Vehicle average speed	90 km/h
Transmit power of the vehicle, P_u	23 dBm
Receive power of the vehicle, P_r	26 dBm
Transmit power of the BSs, P_B	30 dBm
Noise power, N_0	-114 dBm
Computation rate of the VEC server, f_c	10 GHz

that the impact of Doppler effects has been compensated for through the enhanced DMRS technology and will not have a significant impact on the simulations. The main parameters used in the simulations are summarized in Table III. Note that some parameters may change in different figures.

We compare the performances of the proposed MDP offloading and fetching scheme with other three online baseline schemes :

- Adventurous scheme [6]. The system never fetches intermediate results at any slot to lower the current energy consumption unless the last task is offloaded. This scheme is widely adopted in the traditional task-indivisible scenario in which tasks computed at the edge are fetched only after they have all been processed.
- Conservative scheme [42]. The system always fetches intermediate results at each slot to avoid extra recomputations as much as possible. This is the traditional scheme in the task-divisible scenario.
- Threshold-based scheme [43]. In the scheme, the system weighs offloading and fetching to balance the energy consumption of recomputation and frequent fetches. Specifically, considering that the negative fetching decisions may incur more potential energy consumption due to the retransmission of recomputed tasks, the future recomputation energy cost is reflected into the current cost to guide the current decision, referred to as risk cost $C_t^r(\mathbf{s}, a)$. The system makes an online decision to minimize the weighted cost, i.e., $C_t(\mathbf{s}, a) = \beta E_t'(\mathbf{s}, a) + (1-\beta)\, C_t^r(\mathbf{s}, a)$, where $E_t'(\mathbf{s}, a)$ is an immediate communication cost defined in (26). We set $\beta = 0.5$ in the simulation. Additionally, if the last task is offloaded, the system will fetch its results. This scheme can be written in the form of a threshold-based,

$$\pi_t(\mathrm{s}) = \begin{cases} 1, & C_t^r(\mathrm{s}, 1) < \mathcal{E}_t(\mathrm{s}) \\ 2, & C_t^r(\mathrm{s}, 1) > \mathcal{E}_t(\mathrm{s}) \mid\mid \mathrm{s}(3) = M. \end{cases} \tag{34}$$

where the $\mathcal{E}_t(\mathrm{s})$ is the dynamic threshold, $\mathcal{E}_t(\mathrm{s}) = \frac{\beta}{1-\beta}\left(E_t'(\mathrm{s}, 2) - E_t'(\mathrm{s}, 1)\right)$.

In the above three schemes, an additional constraint should be imposed on the decision-making. Considering that the delay under all state-action pairs is available before a decision is made, the standby decision ($a = 0$) will replace the offloading or fetching decision ($a = 1, 2$) to avoid invalid energy consumption when the offloading or fetching operation is predicted to time out.

For comparison, we also figure out the lower bound of the total energy consumption expectation. We consider an offline case [31] and thus $\mathcal{J}^\pi$ can achieve global optimum because the cost function E_t' and transition function P_t' at each slot are revealed. For optimal offline decisions, the system picks optimal slots for offloading or fetching from all the given slots so that $\mathcal{J}^\pi$ is minimized. The classic backward induction (BI) algorithm [44] is adopted to determine the optimal policy. The algorithm consists of the following three steps:

1) the algorithm first sets $V_T(\mathrm{s}) = \min_a E_T'(\mathrm{s}, a)$ for all $\mathrm{s} \in \mathbf{S}_{\text{end}}$ and $V_T(\mathrm{s}) = \mathrm{H}$ for all $\mathrm{s} \in \mathbf{S}_{\text{n-end}}$ to avoid a selfish offloading policy as (31) does.
2) the algorithm computes $V_t(\mathrm{s}), \mathrm{s} \in \mathbf{S}$ in reverse chronological order from $t = T-1$ to $t = 1$ based on the equation (28),
3) the algorithm determines the optimal policy according to (29) for $t = 1, \cdots, T-1$ and the optimal policy for $t = T$ is $\pi_T^*(\mathrm{s}) = \arg\min_{a \in \mathbf{A}} E_t'(\mathrm{s}, a)$.

For the above online and offline policies π in a movement trajectory, the expectation of the total energy consumption $\mathcal{J}^\pi$ is obtained by iteratively updating the following Bellman Equation [35] in reverse chronological order from $t = T-1$ to $t = 1$,

$$V_t(\mathrm{s}) = E_t'(\mathrm{s}, \pi_t(\mathrm{s})) + \sum_{\mathrm{s}' \in \mathbf{S}} P_t'\left(\mathrm{s}' \mid \mathrm{s}, \pi_t(\mathrm{s})\right) V_{t+1}\left(\mathrm{s}'\right), \tag{35}$$

where $V_T(\mathrm{s})$ equals to $E_T'(\mathrm{s}, \pi_T(\mathrm{s}))$. The $\mathcal{J}^\pi$ in a simulation trajectory is equivalent to $V_1(\mathrm{s}_0)$, where s_0 is an initial state fixed to $(1, 1, 1, 0)$, and $\mathcal{J}^\pi$ is averaged over 1000 simulation trajectories in the simulation.

A. Performance Analysis of Proposed OVI for Different Task Parameters

In order to evaluate the performance of the proposed OVI scheme in different task scenarios, three task parameters are examined, i.e., the task size, the number of tasks, and the ratio of output size to task size.

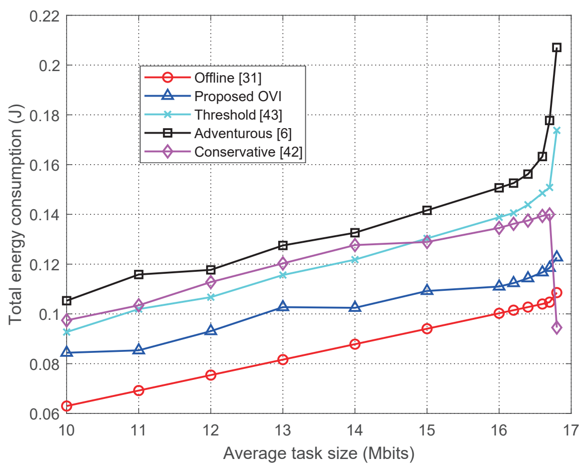


Fig. 7. The total energy consumption in different task sizes.

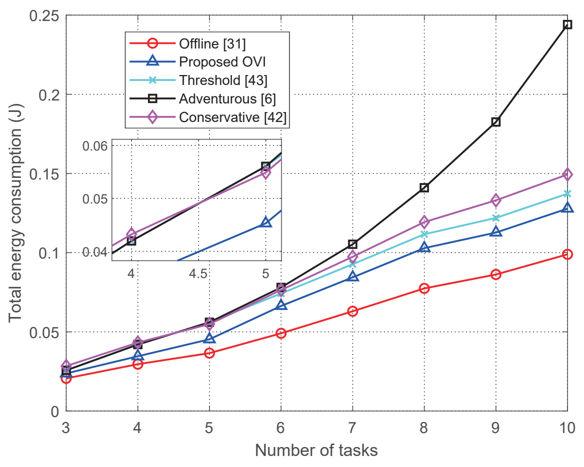


Fig. 8. The total energy consumption in the different number of tasks.

1) Impact of the task size: The total energy consumption $\mathcal{J}^{\pi}$ of the five schemes for different task sizes is shown in Fig. 7. With the increase in the average task size, the total energy consumption also gradually increases. This is because when the task size increases, the transmission delay increases, thereby incurring more energy consumption. Furthermore, our OVI scheme is superior to other online schemes with lower total energy consumption for different task sizes. The superiority of our scheme lies in the long-term consideration of the energy costs in a real-time environment, and the offloading and fetching policy for each slot is determined by minimizing the state value function (30) that takes into account the state transition probabilities derived from the vehicle mobility model and the acquired real-time transmission rates, rather than by myopic optimization of each slot.

As the task size approaches the limit of what can be successfully transferred within T_0, the number of slots available for successful offloading decreases. In this case, our OVI scheme, similar to the offline scheme, basically maintains steady growth. In contrast, the performances of the threshold-based scheme and the adventurous scheme are worse, requiring frequent recomputations and sharply increasing energy consumption due to the increased likelihood of moving out as the task completion period is lengthened. Note that the performance of the conservative scheme in the limit case is even worse because the vehicle does not offload tasks if it cannot fetch the intermediate results, therefore decreasing the number of completed tasks during T slots, which accounts for the significant reduction in energy consumption in Fig. 7.

2) Impact of the number of tasks: The total energy consumption for the different numbers of tasks is shown in Fig. 8. The total energy consumption of our proposed scheme increases more slowly as the number of tasks M increases, compared with other online schemes. Specifically, the possibility of completing all tasks within a single BS decreases as the number of tasks increases, therefore significantly increasing the energy consumption imposed by frequent move-outs and recomputations for the adventurous scheme. For the conservative scheme, frequent fetches incur greater energy consumption in downlink transmission for a small number of tasks, while more recomputation overhead is avoided at a relatively small fetch cost for a large number of tasks. Therefore, the consumption of the conservative scheme is lower than that of the adventurous scheme from $M = 5$. The threshold-based scheme avoids blind offloading or fetching by compromising in recomputation and communication to a certain extent. However, it generates an energy loss by focusing excessively on whether to fetch and neglecting the selection of the optimal offloading slots. Furthermore, the superiority of our OVI scheme is due to the joint consideration of the immediate communication cost (the first term in (30)) and the potential recomputation cost in the real-time environment (the second term in (30)), and the policy is determined by minimizing the joint costs.

3) Impact of the ratio of output size to task size: At last, we evaluate the performance of OVI schemes for different ratios of output size to task size. We consider a general case where the size of output results w_m is smaller than the task size L_m, and set the upper limit of the ratio to 50% to avoid the impact of migration timeouts on the experiment due to oversized migration results. As can be seen from Fig. 9, our proposed scheme outperforms other online schemes at all ratios, that is, our scheme is adapted to various task scenarios with different output ratios. This is because our scheme takes the fetch energy consumption into account in the cost function (24), and thus adaptively reduces the number of fetches when the fetch energy consumption is large (corresponding to a large output size). It also explains why the consumption of our proposed OVI scheme gradually approaches that of the adventurous scheme as the ratio increases.

B. Performance Analysis of Proposed OVI for Different Numbers of BSs

In Fig. 10, we show the total energy consumption performance of the proposed OVI scheme for different numbers of BSs. It can be seen that the total energy consumption increases as the number of BSs increases. This is because the move-out probability of the vehicle, or the BS handover probability increases as the BSs are deployed more densely with a fixed average speed of the vehicle, thus incurring a greater energy

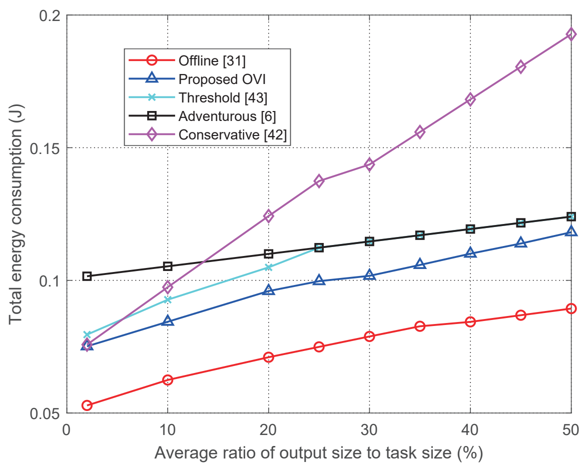


Fig. 9. The total energy consumption in different ratios of output size to task size.

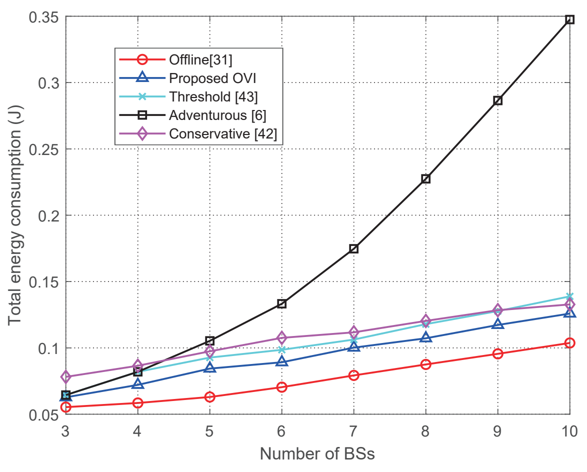


Fig. 10. The total energy consumption in different numbers of BSs.

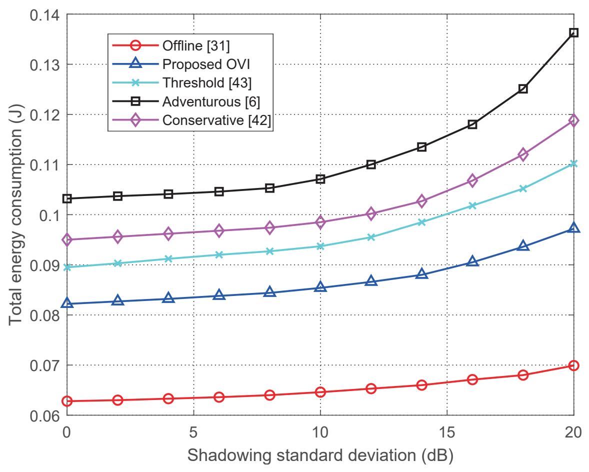


Fig. 11. The total energy consumption in different levels of channel variation.

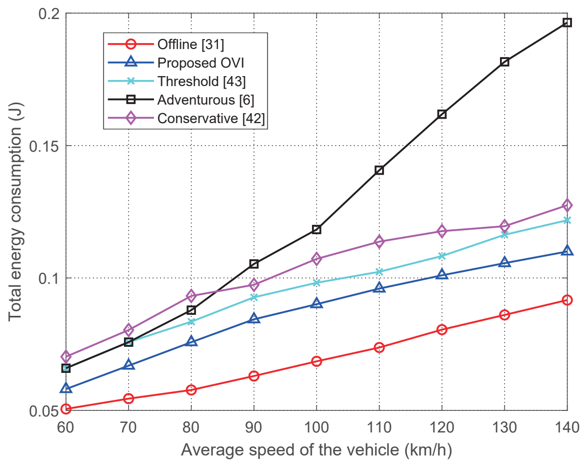


Fig. 12. The total energy consumption at different speeds of the vehicle.

cost of recomputations. Compared to other schemes, our proposed OVI scheme yields lower total energy consumption for different numbers of BSs. This is attributed to the fact that the potential recomputation cost is considered. When the recomputation cost increases due to more frequent handovers, our scheme adaptively increases the number of fetches to reduce the amount of recomputed tasks. However, since the recomputation cost is not considered for the adventurous scheme, the total energy consumption grows approximately exponentially with the increasing number of BSs. In contrast, the conservative scheme and the threshold-based scheme consider recomputation costs but ignore the selection of optimal offload timeslots as stated before. It is concluded that our OVI scheme performs better in the random move-out scenario, especially with dense BSs deployments.

C. *Performance Analysis of Proposed OVI for Different Variation Levels of SNR*

In Fig. 11, we show the total energy consumption performance of the proposed OVI scheme with respect to the variation of signal-to-noise ratio (SNR), considering the highly dynamic characteristics of the channel when the vehicle is moving. We characterize the variation level of SNR in terms of the standard deviation of shadowing from a large-scale propagation effect. Note that we keep the average speed of the vehicle constant to avoid the interference of the variation of move-out probability on the simulation. Also, the effect of path loss on the channel is essentially consistent among experiments due to the fixed mobility model. We can observe that for all the schemes, the total energy consumption is insensitive to the increase in shadowing standard deviation when $\xi < 10$ while it begins to increase when $\xi > 10$. The reason is that, when ξ is small, the channel is lightly fluctuating and can guarantee task transmission and results fetch; when ξ becomes large, the transmission rate may suffer from severe degradation, thus leading to more failures in fetching as well as more recomputed tasks. The gain in transmission rate from channel fluctuation is not sufficient to offset the negative impact of recomputations. Nevertheless, the total energy consumption of our proposed scheme increases at a lower rate than that of the online benchmark schemes, thanks to its efficient failure-aware migration mechanism to balance the fetching energy and recomputation energy.

D. The Performance Analysis of Proposed OVI for Different Speeds of the Vehicle

Fig. 12 shows the influence of the average speed of the vehicle on the total energy consumption. In the scenario of intermittent computation and intermittent communication, the variation of the average vehicle speed induces changes in both the move-out probability as well as the channel. We can see that the total energy consumption increases with the average speed in all the schemes, due to the growth in recomputation tasks. Compared to the other online schemes, our proposed scheme has lower energy consumption at different vehicle speeds, especially in high-speed scenarios, which implies a higher move-out probability and a faster channel change. This is because our scheme takes into account both the fetch cost and the potential recomputation cost, and adaptively adjusts the offload and fetch decisions to channel variations and move-out possibilities by minimizing the joint costs.

VI. Conclution

In this paper, we have studied the problem of dependency-aware computation offloading and service migration under the scenario without backhaul. We have developed a novel dependency-aware indirect migration scheme and have jointly optimized offloading and fetching decisions based on a time-varying MDP model. The general expression of time-varying transition probabilities has been derived to characterize the dynamics of intermittent offloading and fetching under the scenarios of temporally varying vehicular mobility patterns and channel qualities. To solve the MDP problem with both time-varying transitions and immediate costs, we have designed an online algorithm, called OVI, based on an online implementation of value iterations. Taking both immediate communication cost and potential recomputation cost into consideration, our proposed algorithm can optimize energy consumption performance while satisfying task dependency constraints. Simulations have shown that our proposed OVI algorithm can achieve superior energy performance compared to the oracle online schemes, especially when the vehicle mobility is high.

Appendix
Proof of Convergence for the OVI algorithm

The proposed online value iteration (OVI) algorithm can converge to the optimal value at each slot t. We first prove the convergence of a general value iteration algorithm and then prove the convergence of the proposed OVI algorithm based on this convergence theorem.

Theorem 1 (Convergence theorem for value iteration)**.** *For the following value iteration, $U_k(s)$ converges to the unique value $U_*(s)$ when $k\to\infty$.*

$$U_{k+1}(s):=\max_{a\in\mathcal{A}}\sum_{s'\in\mathcal{S}}p\left(s'|s,a\right)\left[r\left(s,a,s'\right)+\gamma U_k\left(s'\right)\right],\gamma\in[0,1). \tag{36}$$

Proof. We first define a Bellman optimality backup operator $\mathcal{B}_*$ as

$$\mathcal{B}_*U(s):=\min_{a\in\mathcal{A}}\sum_{s'\in\mathcal{S}}p\left(s'|s,a\right)\left[r\left(s,a,s'\right)+\gamma U\left(s'\right)\right], \tag{37}$$

where $s\in\mathcal{S}=\{s_1,s_2,\cdots,s_n\}$. Define two vectors $\mathbf{U}_1=[U_1(s_1),U_1(s_2),\cdots,U_1(s_n)]^T$, and $\mathbf{U}_2=[U_2(s_1),U_2(s_2),\cdots,U_2(s_n)]^T$, then

$$\begin{aligned}
&\|\mathcal{B}_*\mathbf{U}_1-\mathcal{B}_*\mathbf{U}_2\|_\infty\\
&=\max_s\Bigg\{\Bigg|\max_{a_1\in\mathcal{A}}\sum_{s'\in\mathcal{S}}p\left(s'|s,a_1\right)\left[r\left(s,a_1,s'\right)+\gamma U_1\left(s'\right)\right]\\
&\qquad-\max_{a_2\in\mathcal{A}}\sum_{s'\in\mathcal{S}}p\left(s'|s,a_2\right)\left[r\left(s,a_2,s'\right)+\gamma U_2\left(s'\right)\right]\Bigg|\Bigg\}\\
&\le\max_s\Bigg\{\max_{a_1\in\mathcal{A}}\Bigg|\sum_{s'\in\mathcal{S}}p\left(s'|s,a_1\right)\left[r\left(s,a_1,s'\right)+\gamma U_1\left(s'\right)\right]\\
&\qquad-\sum_{s'\in\mathcal{S}}p\left(s'|s,a_1\right)\left[r\left(s,a_1,s'\right)+\gamma U_2\left(s'\right)\right]\Bigg|\Bigg\}\\
&\le\gamma\max_s\left\{\max_{a\in\mathcal{A}}\left|\sum_{s'\in\mathcal{S}}p\left(s'|s,a\right)\left[U_1\left(s'\right)-U_2\left(s'\right)\right]\right|\right\}\\
&\le\gamma\max_s\left\{\max_{a\in\mathcal{A}}\left[\sum_{s'\in\mathcal{S}}p\left(s'|s,a\right)\left|U_1\left(s'\right)-U_2\left(s'\right)\right|\right]\right\}\\
&\le\gamma\max_s\left\{\max_{a\in\mathcal{A},s'}\left\{\left|U_1\left(s'\right)-U_2\left(s'\right)\right|\right\}\right\}\\
&=\gamma\left\|\mathbf{U}_1-\mathbf{U}_2\right\|_\infty.
\end{aligned} \tag{38}$$

Therefore, $\mathcal{B}_*$ is proved as a contracting mapping. Define $\mathcal{B}_*^2\mathbf{U}=\mathcal{B}_*\left(\mathcal{B}_*\mathbf{U}\right)$, $\mathcal{B}_*^3\mathbf{U}=\mathcal{B}_*\left(\mathcal{B}_*\left(\mathcal{B}_*\mathbf{U}\right)\right)$, $\cdots$, $\mathcal{B}_*^m\mathbf{U}=\mathcal{B}_*\left(\mathcal{B}_*^{m-1}\mathbf{U}\right)$, then

$$\begin{aligned}
\left\|\mathcal{B}_*^{m+1}\mathbf{U}-\mathcal{B}_*^m\mathbf{U}\right\|_\infty&=\left\|\mathcal{B}_*(\mathcal{B}_*^m\mathbf{U})-\mathcal{B}_*(\mathcal{B}_*^{m-1}\mathbf{U})\right\|_\infty\\
&\le\gamma\left\|\mathcal{B}_*^m\mathbf{U}-\mathcal{B}_*^{m-1}\mathbf{U}\right\|_\infty\\
&\le\gamma^2\left\|\mathcal{B}_*^{m-1}\mathbf{U}-\mathcal{B}_*^{m-2}\mathbf{U}\right\|_\infty\\
&\cdots\\
&\le\gamma^m\left\|\mathcal{B}_*\mathbf{U}-\mathbf{U}\right\|_\infty
\end{aligned} \tag{39}$$

So, with $m\to\infty$, $\lim_{m\to\infty}\left(U_{m+1}-U_m\right)=0$. The sequence $\left\{\mathbf{U},\mathcal{B}_*\mathbf{U},\mathcal{B}_*^2\mathbf{U},\cdots\right\}$ will converge to a fixed value $\mathbf{U}_*$.

The above proof proves the existence of the fixed point, and we next prove the uniqueness by contradiction.

Assuming $\mathbf{U},\mathbf{V}$ are both fixed points with $\mathbf{U}\neq\mathbf{V}$, then,

$$\|\mathcal{B}_*\mathbf{U}-\mathcal{B}_*\mathbf{V}\|_\infty=\|\mathbf{U}-\mathbf{V}\|_\infty, \tag{40}$$

and according to (38), we can conclude that

$$\|\mathcal{B}_*\mathbf{U}-\mathcal{B}_*\mathbf{V}\|_\infty\le\gamma\|\mathbf{U}-\mathbf{V}\|_\infty<\|\mathbf{U}-\mathbf{V}\|_\infty \tag{41}$$

According to the contradiction of (40) and (41), the hypothesis is false, that is, the fixed point V_* is unique.

□

For the proposed OVI algorithm, K iterations of the following value iteration are performed at each slot t.

$$V_t^k(s)=\min_{a\in\mathcal{A}}\left\{E_t'(s,a)+\sum_{s'\in\mathcal{S}}P_t'\left(s'|s,a\right)V_t^{k-1}\left(s'\right)\right\} \tag{42}$$

The value iteration in (42) is a special case of the general value iteration in (36), where the reward function $r(s,a,s')$ is replaced by a negative cost function $-E_t'(s,a)$. Therefore, Theorem 1 guarantees that the algorithm converges to the optimal value as K approaches infinity. In practical simulations, since $p(s' \mid s, a)$ in the algorithm is sparse, meaning that for most state-action pairs (s, a, s'), p is equal to 0, a small value of K can lead to convergence to the optimal value.

References

[1] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi, "Edge computing for autonomous driving: Opportunities and challenges," *Proc. IEEE*, vol. 107, no. 8, pp. 1697–1716, 2019.

[2] K. Zhang, Y. Mao, S. Leng, S. Maharjan, and Y. Zhang, "Optimal delay constrained offloading for vehicular edge computing networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2017, pp. 1–6.

[3] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7944–7956, 2019.

[4] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. ZHANG, "Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading," *IEEE Veh. Technol. Mag.*, vol. 12, no. 2, pp. 36–44, 2017.

[5] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surv. Tutor.*, vol. 19, no. 4, pp. 2322–2358, 2017.

[6] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569–4581, 2013.

[7] C. You, K. Huang, and H. Chae, "Energy efficient mobile cloud computing powered by wireless energy transfer," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1757–1771, 2016.

[8] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, 2017.

[9] C. You and K. Huang, "Exploiting non-causal CPU-state information for energy-efficient mobile cooperative computing," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 4104–4117, 2018.

[10] C.-F. Liu, M. Bennis, M. Debbah, and H. V. Poor, "Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4132–4150, 2019.

[11] L. Tan, Z. Kuang, L. Zhao, and A. Liu, "Energy-efficient joint task offloading and resource allocation in OFDMA-based collaborative edge computing," *IEEE Trans. Wireless Commun.*, vol. 21, no. 3, pp. 1960–1972, 2022.

[12] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *IEEE Trans. Veh. Technol.*, vol. 65, no. 6, pp. 3860–3873, 2016.

[13] J. Du, F. R. Yu, X. Chu, J. Feng, and G. Lu, "Computation offloading and resource allocation in vehicular networks based on dual-side cost minimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1079–1092, 2019.

[14] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4377–4387, 2019.

[15] X. Wang, Z. Ning, S. Guo, and L. Wang, "Imitation learning enabled task scheduling for online vehicular edge computing," *IEEE Trans. Mob. Comput.*, vol. 21, no. 2, pp. 598–611, 2022.

[16] J. Liu, J. Wan, B. Zeng, Q. Wang, H. Song, and M. Qiu, "A scalable and quick-response software defined vehicular network assisted by mobile edge computing," *IEEE Commun. Mag.*, vol. 55, no. 7, pp. 94–100, 2017.

[17] K. Hejja, S. Berri, and H. Labiod, "Network slicing with load-balancing for task offloading in vehicular edge computing," *Veh. Commun.*, vol. 34, p. 100419, 2022.

[18] S. Wang, J. Xu, N. Zhang, and Y. Liu, "A survey on service migration in mobile edge computing," *IEEE Access*, vol. 6, pp. 23 511–23 528, 2018.

[19] Z. Rejiba, X. Masip-Bruin, and E. Marín-Tordera, "A survey on mobility-induced service migration in the fog, edge, and related computing paradigms," *ACM Comput. Surv.*, vol. 52, no. 5, sep 2019.

[20] T. Taleb, A. Ksentini, and P. A. Frangoudis, "Follow-me cloud: When cloud services follow mobile users," *IEEE Trans. on Cloud Comput.*, vol. 7, no. 2, pp. 369–382, 2019.

[21] S. Wang, R. Urgaonkar, T. He, M. Zafer, K. Chan, and K. K. Leung, "Mobility-induced service migration in mobile micro-clouds," in *Proc. IEEE Mil. Commun. Conf.*, 2014, pp. 835–840.

[22] Q. Yuan, J. Li, H. Zhou, T. Lin, G. Luo, and X. Shen, "A joint service migration and mobility optimization approach for vehicular edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 9041–9052, 2020.

[23] H. Wang, T. Lv, Z. Lin, and J. Zeng, "Energy-delay minimization of task migration based on game theory in MEC-assisted vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 71, no. 8, pp. 8175–8188, 2022.

[24] N. Omheni, I. Bouabidi, A. Gharsallah, F. Zarai, and M. S. Obaidat, "Smart mobility management in 5G heterogeneous networks," *IET Netw.*, vol. 7, no. 3, pp. 119–128, 2018.

[25] S. Son, J. Lee, Y. Park, Y. Park, and A. K. Das, "Design of blockchain-based lightweight V2I handover authentication protocol for VANET," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 3, pp. 1346–1358, 2022.

[26] T. M. Nguyen, A. Yadav, W. Ajib, and C. Assi, "Resource allocation in two-tier wireless backhaul heterogeneous networks," *IEEE Trans. Wireless Commun.*, vol. 15, no. 10, pp. 6690–6704, 2016.

[27] Q.-V. Pham, L. B. Le, S.-H. Chung, and W.-J. Hwang, "Mobile edge computing with wireless backhaul: Joint task offloading and resource allocation," *IEEE Access*, vol. 7, pp. 16 444–16 459, 2019.

[28] W. Fan, J. Liu, M. Hua, F. Wu, and Y. Liu, "Joint task offloading and resource allocation for multi-access edge computing assisted by parked and moving vehicles," *IEEE Trans. Veh. Technol.*, vol. 71, no. 5, pp. 5314–5330, 2022.

[29] Z. Liang, Y. Liu, T.-M. Lok, and K. Huang, "A two-timescale approach to mobility management for multicell mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 21, no. 12, pp. 10 981–10 995, 2022.

[30] J. Yu, Z. Chen, Y. Zhu, Y. Chen, L. Kong, and M. Li, "Fine-grained abnormal driving behaviors detection and identification with smartphones," *IEEE Trans. Mob. Comput.*, vol. 16, no. 8, pp. 2198–2212, 2017.

[31] X. Zhang, J. Zhang, Z. Liu, Q. Cui, X. Tao, and S. Wang, "MDP-based task offloading for vehicular edge computing under certain and uncertain transition probabilities," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 3296–3309, 2020.

[32] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, 2016.

[33] Q. Shen, B.-J. Hu, and E. Xia, "Dependency-aware task offloading and service caching in vehicular edge computing," *IEEE Trans. Veh. Technol.*, 2022.

[34] L. Liang, G. Y. Li, and W. Xu, "Resource allocation for D2D-enabled vehicular communications," *IEEE Trans. Commun.*, vol. 65, no. 7, pp. 3186–3197, 2017.

[35] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[36] Y. Li, A. Zhong, G. Qu, and N. Li, "Online markov decision processes with time-varying transition probabilities and rewards," in *ICML workshop on Real-world Sequential Decision Making*, 2019.

[37] D. N. Alparslan and K. Sohraby, "A generalized random mobility model for wireless ad hoc networks and its analysis: One-dimensional case," *IEEE ACM Trans Netw*, vol. 15, no. 3, pp. 602–615, 2007.

[38] G. C. and R. L. B., *Statistical inference*. Cengage Learning, 2021.

[39] C. Shu, Z. Zhao, Y. Han, G. Min, and H. Duan, "Multi-user offloading for edge computing networks: A dependency-aware and latency-optimal approach," *IEEE Internet Things J.*, vol. 7, no. 3, pp. 1678–1689, 2019.

[40] 3rd Generation Partnership Project (3GPP), "Technical specification group radio access network: Study LTE-based V2X services: (release 14)," Standard 3GPP TR 36.885 V2.0.0, Jun. 2016.

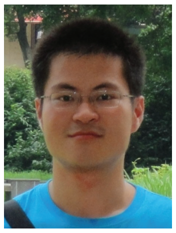
[41] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, 2016.

[42] S. E. Mahmoodi, R. N. Uma, and K. P. Subbalakshmi, "Optimal joint scheduling and cloud offloading for mobile applications," *IEEE Trans. on Cloud Comput.*, vol. 7, no. 2, pp. 301–313, 2019.

[43] D. Wang, X. Tian, H. Cui, and Z. Liu, "Reinforcement learning-based joint task offloading and migration schemes optimization in mobility-aware MEC network," *China Commun.*, vol. 17, no. 8, pp. 31–44, 2020.

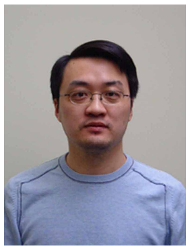
[44] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Qibing Fan received the B.E. degree in electronic and information engineering from Nanjing University of Science and Technology (NJUST), Nanjing, China, in 2021. He is currently pursuing the M.S. degree with the Department of Electronic Engineering and Information Science, University of Science and Technology of China. His research interests include edge computing and the Internet of Vehicles.



Li Chen (Senior Member, IEEE) received the B.E. degree in electrical and information engineering from the Harbin Institute of Technology, Harbin, China, in 2009, and the Ph.D. degree in electrical engineering from the University of Science and Technology of China, Hefei, China, in 2014. He is currently an Associate Professor with the Department of Electronic Engineering and Information Science, University of Science and Technology of China. His research interests include integrated communication and computation, integrated sensing and communication, and wireless IoT networks.

Changsheng You (Member, IEEE) received his B.Eng. degree in 2014 from University of Science and Technology of China (USTC) and Ph.D. degree in 2018 from The University of Hong Kong (HKU). He is currently an Assistant Professor at Southern University of Science and Technology, and was a Research Fellow at National University of Singapore (NUS). His research interests include intelligent reflecting surface, UAV communications, edge learning, mobile-edge computing. Dr. You is an editor for IEEE Communications Letters (CL), IEEE IEEE Transactions on Green Communications and Networking (TGCN), and IEEE Open Journal of the Communications Society (OJ-COMS). He received the IEEE Communications Society Asia-Pacific Region Outstanding Paper Award in 2019, IEEE ComSoc Best Survey Paper Award in 2021, IEEE ComSoc Best Tutorial Paper Award in 2023. He is listed as the Highly Cited Chinese Researcher, Exemplary Reviewer of the IEEE Transactions on Communications (TCOM) and IEEE Transactions on Wireless Communications (TWC).



Yunfei Chen (Senior Member, IEEE) received the B.E. and M.E. degrees in electronics engineering from Shanghai Jiaotong University, Shanghai, China, in 1998 and 2001, respectively, and the Ph.D. degree from the University of Alberta in 2006. He is currently a Professor with the Department of Engineering, University of Durham, U.K. His research interests include wireless communications, cognitive radios, wireless relaying, and energy harvesting.

Huarui Yin (Member, IEEE) received the B.E. and Ph.D. degrees in electronic engineering and information science from the University of Science and Technology of China (USTC), Hefei, Anhui, in 1996 and 2006, respectively. Since 2010, he has been with the Department of Electronic Engineering and Information Science, USTC, as an Associate Professor. His research interests include digital signal processing, low-complexity receiver design, multiple access for massive connections, and the throughput analysis of wireless networks.