Energy-efficient Caching and Task offloading for Timely Status Updates in UAV-assisted VANETs

Nan Hu*, Xiaqi Qin*, Nan Ma*, Yiming Liu*, Yuanyuan Yao†, Ping Zhang*
*State Key Laboratory of Networking and Switching Technology,
Beijing University of Posts and Telecommunications, Beijing, China
†Beijing Information Science and Technology University, Beijing, China
Email:{nanhu, xiaoqiqin, liuyiming, manan, pzhang}@bupt.edu.cn, yyyao@bistu.edu.cn

Abstract—Intelligent edge network is maturing to enable smart and efficient transportation systems. In this paper, we consider UAV-assisted vehicular networks where UAVs provide caching and computing services in complement with BS. One major challenge is that vehicles need to obtain timely situational awareness via orchestration of ubiquitous caching and computing resources. Note that cached data for vehicles’ perception tasks contains time-varying context information, thus freshness of cached data should be considered in conjunction with task execution to guarantee timeliness of obtained status updates. To this end, we propose a two-stage performance metric to quantify the impact of cache refreshing and computation offloading decisions on the age of status updates. We formulate an energy minimization problem by jointly considering cache refreshing, computation offloading and aging of status updates. To facilitate online decision making, we propose a deep deterministic policy gradient (DDPG)-based solution procedure and incorporate differentiated experience replay mechanism to accelerate convergence. Simulation results show that the performance of proposed solution is competitive in terms of energy consumption for obtaining fresh status updates.

Index Terms—Age of Information, Mobile Edge Computing (MEC), Edge Caching, Deep Reinforcement Learning

I. INTRODUCTION

The advent of edge intelligence empowered 5G networks unlocks the potential for automotive industries, such as smart transportation system [1]. To reduce human intervention in critical operations such as path planning and obstacle avoidance, it’s essential for autonomous vehicles to obtain real-time situational awareness of surroundings via intelligent orchestration of ubiquitous caching and computing capabilities [2].

This new paradigm involves information flows around a control-loop from the vehicle to edge server and back to the vehicle. For example, a vehicle continuously generates environment perception tasks and offloads to edge server in short of on-board computing and caching resources. Then edge server executes the task (e.g., simultaneous localization and mapping) based on its cached data (e.g., pre-built HD maps) and feeds status updates back to the vehicle. Note that the cached data usually contains dynamic driving-related context information, which should be refreshed frequently. Therefore, the timeliness of obtained status updates is determined by task execution duration and freshness of cached computing data.

Owing to the prompt deployment, unmanned aerial vehicles (UAVs) are widely used to complement edge server by providing additional computing capability for vehicular networks from bird’s-eye view. In most existing works [4], the consecutively generated computation tasks are treated independently, and the offloading strategy focuses on minimizing one-shot task execution delay. However, as for time-critical control scenarios considered in this paper, status updates obtained by executing computation tasks are temporally correlated. Moreover, a stale status update is of less value in terms of the degree to which it represents reality. In this sense, we employ the concept of age of information (AoI) to quantify the timeliness of obtained status updates [5], where a larger value of age indicates that the status update is of less value for the accuracy of environment perception. The concept of AoI has been applied as a performance metric in computation offloading [6] or cache deployment [7], separately. In [6], AoI is applied in an edge computing enabled remote monitoring system to quantify the time elapsed since the newest processed status update is sampled at source, which serves as the objective function to jointly optimize status sampling and computation offloading. In [7], AoI is generalized as a binary function to keep track of instantaneous freshness of latest cached versions of files, which guides the design of optimal cache updating rates.

Different from these existing works, in this paper, we aim to investigate a closed-loop optimization for obtaining timely status updates by proposing a two-stage AoI metric to capture the intertwined relationship between cache refreshing and computation offloading, so as to strike a balance between the timeliness of obtained status updates and the required energy budget. More specifically, we consider an UAV-assisted vehicular network, where UAVs are deployed to provide flexible caching and computing services in complement with base station (BS). To obtain timely status updates in an energy-efficient manner, we develop an energy minimization problem by jointly considering cache refreshing, task offloading, and aging of status updates at vehicles. In the formulated problem, we characterize the temporal correlations among task generation, task execution and cache refreshing, which serves as the measure of age of obtained status updates. To achieve real-time decision making, we propose a differentiated experience replay based DDPG algorithm. Simulation results validate the effectiveness of our proposed solution.
II. MODELING AND PROBLEM FORMULATION

A. Scenario Description

Consider an UAV-assisted vehicular network as shown in Fig. 1, which consists of a base station, a set of $N$ vehicles and a set of $M$ UAVs loitering over a specific segment of street with a constant speed and altitude. Denote $N = |N|$ as the number of vehicles and $M = |M|$ as the number of UAVs. Each UAV provides flexible cache-enabled edge computing service for vehicles within its coverage area, which caches repetitively requested input data (e.g., high-resolution map) for timely task execution. Denote $\mathcal{W}$ as types of computation tasks. Upon each arrival of computation task $w$ at a vehicle, the task execution can be accomplished in three ways: executed locally using data cached at a vehicle, offloaded to a nearby UAV using the “flying” cache, or offloaded to base station which caches the most up-to-date data for all types of tasks. Note that the size of cache at vehicles and UAVs is limited, thus the cached data has to be proactively selected and refreshed to collaborate with the task offloading decision. Table I lists notation used in this paper.

![Fig. 1: An UAV-assisted vehicular network.](image)

| Symbol       | Definition                                                                 |
|--------------|---------------------------------------------------------------------------|
| $N$          | The set of vehicles                                                      |
| $M$          | The set of UAVs                                                           |
| $\mathcal{W}$| The set of cached data                                                   |
| $I$          | The set of time slots                                                    |
| $R$          | Transmission range of UAV                                                |
| $H$          | Flying height of UAV                                                     |
| $P_{tr}$     | Transmission power at vehicle $i$                                        |
| $\tau$       | The length of a time slot                                                |
| $i_w$        | The size of cached data $w$                                               |
| $A_{i,w}^t$  | The age of cached data $w$ at vehicle $i$ at time slot $t$                |
| $A_{i,w}^t$  | The age of cached data $w$ at UA V at time slot $t$                      |
| $A_i(t)$     | The age of status updates of vehicle $i$ at time slot $t$                 |
| $C_{veh}$    | The storage space at vehicle $i$                                         |
| $C_{uav}$    | The storage space at UA V                                                |
| $\mu$        | Energy coefficient per CPU cycles                                       |
| $\gamma$     | Computing resources allocated to vehicle $i$ at UAV $j$ or BS $w$        |
| $\rho$       | Energy consumption for data fetching                                     |
| $y_{i,w}^t$  | Task execution duration at UA V $j$ or BS $w$ at time slot $t$           |
| $A_{i,w}^t$  | Task execution duration from vehicle $i$ to UAV $j$ or BS $w$ at time slot $t$ |
| $\beta_{i,j}(t)$ | Channel gain between vehicle $i$ and UAV $j$                             |

B. Timeliness of Status Updates

Assume that sample-at-change strategy is adopted, where each vehicle continuously generates and executes tasks to capture changes of status about its surroundings. Denote $\hat{A}_i(t)$ as age of status updates of vehicle $i$ at time slot $t$, which characterizes how much uncertainty can be eliminated by obtaining a status update at time slot $t$ via task execution. Note that a smaller value of age indicates that the obtained status update is fresher. Then if a task is executed at time slot $t$, the age of obtained status update ($\hat{A}_i(t)$) is determined by i) system time of the first unprocessed task in buffer (i.e., time duration from the task is generated until it is executed), and ii) freshness of cached data for task execution. Therefore, it involves two sets of decisions: where tasks are executed (locally at vehicle, offloaded to UAV or base station) and to refresh the cached data at vehicles and UAVs?

To characterize the impact of aforementioned decision variables on timeliness of status updates, we propose two-stage performance metrics (i.e., age of cached data and age of status updates) based on the concept of age of information (AoI) [5]. Note that since base station caches the most up-to-date data, the age of cached data is always 0. Denote $A_{i,w}^t$ as age of cached data for task $w$ at vehicle $i$ at time slot $t$, as shown in Fig. 2, the cache refreshing has four possibilities: i) $A_{i,w}^t$ starts at 1 when it is newly added to cache; ii) $A_{i,w}^t$ increases linearly with $t$ if it is not updated; iii) $A_{i,w}^t$ drops to 1 if it is updated; iv) $A_{i,w}^t$ jumps to infinity if it is deleted due to limited storage space. Then we have:

\[
A_{i,w}^t(t+1) = \begin{cases} 
1, & \text{newly add or update}, \\
I_{\infty} + A_{i,w}^t(t), & \text{delete}, \\
A_{i,w}^t(t) + 1, & \text{otherwise}.
\end{cases}
\]

Denote $y_{i,w}^t$ as a binary variable to indicate whether or not data for task $w$ is cached at vehicle $i$ at time slot $t$. Denote $y_{i,w}^t$ as a binary variable to indicate whether or not a data for task $w$ is updated or newly added to cache at vehicle $i$:

\[
y_{i,w}^t \leq c_{i,w}^t(t+1), \quad (i \in \mathcal{N}, w \in \mathcal{W}).
\]

Then constraints (1) can be transformed as follows:

\[
A_{i,w}^t(t+1) = c_{i,w}^t(t+1)[y_{i,w}^t(t)] + (1 - y_{i,w}^t(t))(A_{i,w}^t(t) + 1) + (1 - c_{i,w}^t(t+1))I_{\infty}, \quad (i \in \mathcal{N}, w \in \mathcal{W}).
\]

Similarly, as for cache refreshing at UAV $j$, we have:

\[
y_{j,w}^t \leq c_{j,w}^t(t+1), \quad (j \in \mathcal{M}, w \in \mathcal{W}).
\]

Then age of status updates ($\hat{A}_i(t)$) can be obtained as shown in Fig. 2. Before the first task is executed, vehicle has no
knowledge about its surroundings and thus $\hat{A}_i(t)$ is set as infinity. The first task is generated at $t_{i,1}^{g}$, once task 1 is offloaded to UAV $j$ and executed at $t_{i,1}$, $A_i(t)$ is reset to the sum of task 1’s system time ($t - t_{i,1}^{g}$) and age of cached data for task 1 at UAV $j$ ($A_{i,j}^{veh}(t)$). Then $\hat{A}_i(t)$ increases linearly with $t$ before task 2 is executed. Denote $t_{i,w}^{g}$ as generation time of task $w$ at vehicle $i$, then we have:

$$\hat{A}_i(t) = \begin{cases} t - t_{i,w}^{g} + A_{i,w}^{veh}(t), & \text{local execution,} \\ t - t_{i,w}^{g} + A_{j,w}^{veh}(t), & \text{offloaded to UAV } j, \\ t - t_{i,w}^{g}, & \text{offloaded to BS}, \\ \hat{A}_i(t-1) + 1, & \text{otherwise.} \end{cases}$$  

(6)

Based on constraints (6), cached data at UAVs and vehicles should be refreshed as frequently as possible to improve the timeliness of status updates. However, frequent cache refreshing brings extra energy consumption at these mobile terminals. Denote $\theta$ (in J/bit) as the energy consumption for data fetching [8]. Denote $\xi(t)$ as system-level energy consumption for cache refreshing at time slot $t$, then we have:

$$\xi(t) = \left[ \sum_{i=1}^{N} \sum_{w=1}^{W} y_{i,w}^{veh}(t)l_w \right] + \left[ \sum_{j=1}^{M} \sum_{w=1}^{W} y_{j,w}^{uav}(t)l_w \right] \cdot \theta. \quad (10)$$

D. Computation Task Execution

Denote $W_i(t)$ as the types of computation tasks in vehicle $i$’s task buffer at time slot $t$. Note that only one task of the same type will be stored in the buffer. As task division is not considered here, a task can be executed in one of three ways: locally at vehicle, offloaded to base station or a nearby UAV.

At each time slot $t$, vehicle $i$ can choose one of three modes: local execution, task offloading, or idle (in case when there is no buffered task). Denoted $x_{i,w}^{loc}(t)$ as a binary variable to indicate whether or not the task is executed locally at vehicle $i$. Denote $x_{i,w}^{veh}(t)$ as a binary variable to indicate whether or not vehicle $i$ offloads the task. Specifically, if $j = M + 1$, it indicates whether vehicle $i$ offloads the task to BS; if $j \leq M$, it indicates whether vehicle $i$ offloads the task to UAV $j$. Then we have:

$$x_{i,w}^{loc}(t) + \sum_{j=1}^{M+1} x_{i,j}^{mec}(t) \leq 1, (i \in N). \quad (11)$$

Note that task execution decisions and cache refreshing decisions are intertwined, since a task can be executed only when its input data has been cached. As for the first unprocessed task $w$ at vehicle $i$ ($w = \max \{ k \mid t - t_{i,k}^{g} \leq T \}$), we have:

$$x_{i,w}^{loc}(t) \leq c_{i,w}^{veh}(t), (i \in N). \quad (12)$$

$$x_{i,j}^{mec}(t) \leq c_{i,j}^{uav}(t), (i \in N, j \in M). \quad (13)$$

Local Execution: In the case when vehicle $i$ executes its first unprocessed task $w$ locally ($w = \max \{ k \mid t - t_{i,k}^{g} \leq T \}$), denote $t_{i,w}^{loc}$ (in cycles/s) as computation capability of vehicle $i$. We assume that task execution must be completed within one time slot. Denote $z_{w}$ as the required number of cycles for task $w$, while $\tau$ represents slot length, then we have:

$$x_{i,w}^{loc}(t) \cdot \frac{z_{w}}{t_{i,w}^{loc}} \leq \tau, (i \in N). \quad (14)$$

Denote $E_{i,w}^{loc}(t)$ as the corresponding energy consumption, and $\mu$ as the energy coefficient per CPU cycle[9], we have:

$$E_{i,w}^{loc}(t) = \mu \cdot \left( t_{i,w}^{loc} \right)^2 \cdot z_{w}, (i \in N). \quad (15)$$

Task offloading: In the case when vehicle $i$ offloads its first unprocessed task to UAV or BS, denote $t_{i,j}^{tr}(t)$ as transmission duration and $t_{i,j}^{loc}(t)$ as execution duration. Assume that task processing must be completed within one time slot. We have:

$$x_{i,j}^{mec} \cdot \left[ t_{i,j}^{loc}(t) + t_{i,j}^{tr}(t) \right] \leq \tau, (i \in N, j \in M). \quad (16)$$
As for execution duration $t^e_{i,j}(t)$, denote $f_{i,j}(t)$ (in cycles/s) as the computing resources allocated to vehicle $i$ at UAV $j$ or BS ($j = M + 1$), while $F_{i,j}^{\text{max}}$ represents the total CPU frequency. Assume that the computing resources are equally divided among the offloaded tasks, then we have:

$$f_{i,j}(t) \leq \frac{F_{i,j}^{\text{max}}}{N}, (i \in \mathcal{N}, 1 \leq j \leq M + 1).$$ (17)

$\text{As for transmission duration } t^r_{i,j}(t), \text{ denote } b_{i,j}(t) \text{ as the bandwidth allocated to vehicle } i, \text{ while } B \text{ represents the total bandwidth available in the system. Then we have:}$

$$t^r_{i,j}(t) = \sum_{j=1}^{M+1} x^\text{mec}_{i,j}(t) \frac{z_w}{b_{i,j}(t)}, (i \in \mathcal{N}).$$ (18)

If vehicle $i$ offloads its task to BS (i.e., $x^\text{mec}_{i,M+1}(t) = 1$), the achievable task transmission rate can be obtained as:

$$r^{\text{off}}_{i,j}(t) = b_{i,j}(t) \log_2 (1 + \frac{P^t_i g_i(t)}{\sigma^2}), (i \in \mathcal{N}, j= M + 1).$$ (20)

where $P^t_i$ is transmission power, $g_i(t)$ is the average channel gain between vehicle $i$ and BS, and $\sigma^2$ is noise power.

Only if vehicle $i$ is within the coverage of UAV $j$, it can offload its task to UAV. At each time slot $t$, given current coordinates of vehicle $i$ ($n^x_i(t), n^y_i(t)$), vehicle $i$’s speed $v^{\text{veh}}_i(t)$ (moving right is the positive direction), coordinates of UAV $j$ ($u^x_j(t), u^y_j(t)$), UAV $j$’s speed $v^{\text{uav}}_j(t)$, altitude $H$ and the radius of UAV’s projected coverage area $R$, the feasibility of offloading to UAV $j$ can be obtained by:

$$d_{i,j}(t+1) = \sqrt{(n^x_i(t) - u^x_j(t))^2 + (n^y_i(t) - u^y_j(t))^2 + H^2}, (i \in \mathcal{N}, j \in \mathcal{M}).$$ (22)

Assume that channel condition changes across time slots and remains constant within a time slot. At each time slot $t$, vehicle $i$ has LoS view towards UAV $j$ with probability [10]:

$$p_{i,j}^\text{LoS}(t) = \frac{1}{1 + \gamma \exp(-\psi[\zeta_{i,j}(t) - \gamma])}, (i \in \mathcal{N}, j \in \mathcal{M}),$$ (23)

where $\psi$ is a constant. The elevation angle $\zeta_{i,j}(t)$ is calculated based on relative positions of vehicle $i$ and UAV $j$:

$$\zeta_{i,j}(t) = \frac{180}{\pi} \sin^{-1} \left( \frac{H}{d_{i,j}(t)} \right), (i \in \mathcal{N}, j \in \mathcal{M}).$$ (24)

Then the channel gain at time slot $t$ can be obtained as:

$$\beta_{i,j}(t) = \left\{ \begin{array}{ll} \frac{1}{\eta_1} \left( \frac{4\pi f_c d_{i,j}(t) \eta_1}{c} \right)^{-\alpha} , & \text{LoS}, \\ \frac{1}{\eta_2} \left( \frac{4\pi f_c d_{i,j}(t) \eta_2}{c} \right)^{-\alpha} , & \text{NLoS}. \end{array} \right.$$ (25)

Denote $\beta_0 = \left( \frac{4\pi f_c}{c} \right)^{-\alpha}$ as channel gain when the distance is set as 1 meter, then we have:

$$\beta_{i,j}(t) = P^t_{i,j}(t) \frac{1}{\eta_1} \beta_0 (d_{i,j}(t))^{-\alpha} + (1 - P^t_{i,j}(t)) \frac{1}{\eta_2} \beta_0 (d_{i,j}(t))^{-\alpha}, (i \in \mathcal{N}, j \in \mathcal{M}).$$ (26)

Then the achievable task transmission rate can be obtained as:

$$r^{\text{on}}_{i,j}(t) = b_{i,j}(t) \log_2 \left(1 + \frac{P^t_i \beta_{i,j}(t)}{\sigma^2} \right), (i \in \mathcal{N}, j \in \mathcal{M}).$$ (27)

Denote $s_w$ as the size of vehicle $i$’s first unprocessed task $w$, then the task transmission duration can be obtained as:

$$t^t_{i,j}(t) = \sum_{j=1}^{M+1} x^\text{mec}_{i,j}(t) \cdot s_w + x^\text{mec}_{i,M+1}(t) \cdot s_w, (i \in \mathcal{N}).$$ (28)

Then the energy consumption of task offloading can be obtained as:

$$E^{\text{off}}_i(t) = P^{\text{tr}} \cdot t^t_{i,j}(t), (i \in \mathcal{N}).$$ (29)

### E. Problem Formulation

We aim at minimizing the system-level energy consumption (including cache refreshing and computation task execution) over $T$ time slots. Then the problem can be formulated as:

**OPT-P**

$$\min_{S(t)} \sum_{t=1}^{T} \left( \xi(t) + \sum_{i=1}^{N} (E^{\text{loc}}_i(t) + E^{\text{off}}_i(t)) \right)$$

s.t. Timeliness of status updates: (3)(5)(6)(7);

Cache refreshing costs: (2)(4)(8)(9);

Computation task execution: (11)-(28).

In this formulation, $S(t)$ represents the set of all variables, where $y_{i,w}(t)$, $y_{i,w}^{\text{on}}(t)$, $c_{i,w}(t)$, $c_{i,w}^{\text{on}}(t)$, $x^{\text{loc}}_i(t)$ and $x^{\text{mec}}_{i,j}(t)$ are binary variables, $b_{i,j}(t)$ are continuous variables. The formulated problem falls in the form of a mixed integer nonlinear program (MINLP), which is intractable. The optimal solution can be obtained offline only if prior knowledge of all the system state information is known, which is impractical in general. To cope with the stochastic task generation at vehicles and time varying channel conditions, we propose a deep reinforcement learning based algorithm to facilitate fast decision-making.

### III. DRL-BASED CACHING AND TASK OFFLOADING ALGORITHM

In this section, we propose an online decision making approach for UAV-assisted vehicular networks, in which at each time slot $t$, the cache refreshing decisions ($y_{i,w}(t)$, $y_{i,w}^{\text{on}}(t)$, $c_{i,w}(t)$, $c_{i,w}^{\text{on}}(t)$), task execution decisions ($x^{\text{loc}}_i(t)$ and $x^{\text{mec}}_{i,j}(t)$), and bandwidth allocation decisions $b_{i,j}(t)$ are optimized in order to minimize system-level energy consumption.

This can be achieved by transforming the formulated problem OPT-P into a MDP problem, which is defined by a tuple $\{S^c, A^c, T^c, R^c\}$, where $S^c$ is the set of state systems, $A^c$ is set of system actions, $T^c = \{p(s'|s,c)\}$ is the set of
transition probabilities, and $R^c : S^c \times A^c \rightarrow R^c$ is a real-value reward function when the system is at state $s^c(t) \in S^c$ and an action $a^c(t) \in A^c$ is taken. A policy $\pi$ is a mapping from $S^c$ to $A^c$. Then the MDP problem of caching and task offloading is defined as follows.

1) **State ($S^c$):** At time slot $t$, the system state is defined as vehicles’ coordinates, UAVs’ coordinates, age of cached data at vehicles, age of cached data at UAVs and age of status updates, $s^c(t) = \{V_i(t), U_j(t), A_{i,w}^{vec}(t), A_{i,w}^{uav}(t), A_i(t)\}$.

2) **Action ($A^c$):** At time slot $t$, BS needs to make decisions for cache refreshing, task execution and bandwidth allocation, $a^c(t) = \{y_{i,w}^{vec}(t), y_{i,w}^{uav}(t), c_{i,w}^{vec}(t), c_{i,w}^{uav}(t), s_{i,w}^{loc}(t), x_{i,w}^{mc}(t), b_{i,j}(t)\}$.

3) **Reward ($R^c$):** We employ the total energy consumption for cache refreshing and task execution (objective function of OPT-P) as reward function $r^c(t)$, which is defined as:

$$
r^c(t) = \left\{ \begin{array}{ll}
\sum_{i=1}^{N} \xi(t) + \sum_{i=1}^{N} E_{i}^{loc}(t) + E_{i}^{mc}(t) & -P.
\end{array} \right.
$$

(30)

where $f(\cdot)$ is the negative exponential function that acts as a normalization, which can be obtained by constraints (2)-(6), (10)-(21) and (23)-(29). The penalty $P$ consists of constraints (7)-(9) and (22), which prevents age threshold violation, cache overflow, and infeasible task offloading due to mobility of UAVs and vehicles.

Define value of the $k$-th state $s_k^c$ as expected long-term discounted reward under policy $\pi$ starting from $s_k^c$.

$$
V(s_k^c|\pi) = \mathbb{E}_\pi \left[ \sum_{l=1}^{\infty} \gamma^{l-1} r^c(t+l) | s_k^c = s^c(t) \right].
$$

Then the state-action-value function can be obtained as:

$$
Q(s_k^c, a_k^c|\pi) = \mathbb{E}_\pi \left[ r^c(t+1) + \gamma V(s_{k+1}^c)|\pi \right].
$$

(32)

Consider continuous state $s^c$ and action $a^c$, we define the following performance objective under a certain policy $\pi$.

$$
J(\pi) = \mathbb{E}_\pi \left[ Q(s^c, a^c|\pi) \right]
= \int_{s^c} ds^c \int_{A^c} da^c \pi(a^c|s^c) Q(s^c, a^c|\pi) da^c ds^c.
$$

(33)

We propose a differentiated experience replay based DDPG algorithm to facilitate online decision making. Note that our proposed algorithm is problem-customized to make experience replay more efficient to achieve faster learning with better performance. More specifically, experience replay in traditional DDPG algorithms employs uniform sampling at random without considering the quality of experience. Considering the fact that an agent may learn more effectively from some transitions (including failures) than from others [11], we classify the transitions into positive experience and negative experience based on a pre-fixed threshold of reward function values (denoted as $R_{th}$), and replay them in proportion to liberate agents from learning correlated transitions in the exact order they experienced.

To distinguish between positive and negative experience, we employ the lower bound value of reward function during convergence oscillation as the threshold ($R_{th}$). Among total number of $N_{step}$ steps within each episode, we choose the first $N_{ne} = N_{step} \cdot \chi$ steps to perform the aforementioned differentiated experience replay strategy, with $N_{ne} = R_{batch} \cdot \rho$ being the number of sampled negative experience. Through numerous simulation, we found out that the most suitable range of coefficients are $\chi \in (0.05, 0.15)$ and $\rho \in (0, 0.2)$.

**IV. PERFORMANCE EVALUATION**

In this section, we present simulation results to demonstrate the performance of our proposed solution. We consider an UAV-assisted vehicular network where two UAVs with loft height of 40m provide flying caching and computation services for vehicles, while the communication range of an UAV is set as 100m [4]. Assume there are 5 types of tasks ($w = 5$), while task size and its required computation cycles follow uniform distribution with $s_w \in [100, 150]$ Kbits and $z_w \in [1 \times 10^5, 1.5 \times 10^7]$ cycles [2]. The task generation frequency at each vehicle follows zipf distribution. The computation capability at UAV is $3 \times 10^8$ cycles/s [2]. The computation capability at vehicles follow uniform distribution with $f_{loc} \in [4.5 \times 10^8, 5.5 \times 10^8]$ cycles/s. Each vehicle can cache data for one task, while UAV can cache data for three tasks. The transmission power at vehicle is 1 W [4]. The energy coefficient of cache fetching is $10^{-8}$ J/bit [8], while energy coefficient for computing is $10^{-27}$ [9]. As for learning parameters, the capacity of experience replay buffer is 10000 [4], which is equally divided into positive and negative buffers. $\chi$ and $\rho$ are set as 0.1.

Fig. 3 compares the learning performance of our proposed solution with traditional DDPG algorithm. As shown in the figure, our proposed solution achieves a better performance with faster convergence rate. It verifies the benefit of our proposed problem-customized differentiated experience replay. The training process takes 120 seconds in average, and once the framework is trained properly, it takes 16 milliseconds to obtain a solution. To demonstrate the performance benefit of our proposed solution, we employ four benchmarks. As for

**Fig. 3: Reward values during training.**
allocation optimization. which verifies the importance of a rigorous design of resource allocation decisions, as well as the necessity of optimizing system resource allocations.

V. CONCLUSION

In this paper, we investigated an energy-efficient caching and task offloading strategy in UAV-assisted vehicular networks. To quantify the timeliness of obtained updates, we employed the concept of age of information to bridge the gap between caching refreshing and task execution. We formulated an energy consumption minimization problem by jointly considering cache refreshing, task execution and bandwidth allocation decisions. To realize fast decision making under stochastic task generations, we proposed a differentiated experience replay based DDPG algorithm. Simulation results demonstrated the performance benefit of our proposed solution in terms of energy efficiency and timeliness of status updates.

ACKNOWLEDGMENTS

This paper is supported by Young Elite Scientists Sponsorship Program by CIC No. 2021QNRC001, National Natural Science Foundation for Young Scientists of China under Grant No.62001050 and Beijing Natural Science Foundation Haidian Original Innovation Joint Fund (No.19L2022).

REFERENCES

[1] X. Xu, S. Gao and M. Tao, “Distributed Online Caching for High-Definition Maps in Autonomous Driving Systems,” IEEE Wireless Communications Letters, vol. 10, no. 7, pp. 1390-1394, July. 2021.
[2] Z. Ning, K. Zhang, X. Wang, L. Guo, X. Hu, J. Huang, B. Hu, R. Kwok, “Intelligent Edge Computing in Internet of Vehicles: A Joint Computation Offloading and Caching Solution,” IEEE Transactions on Intelligent Transportation Systems, vol. 22, no. 4, pp. 2212-2225, April. 2021.
[3] Z. Niu, S. Zhou, N. Crespi, “Greening 6G,” in Shaping Future 6G Systems,” IEEE Transactions on Wireless Communications, vol. 20, no. 3, pp. 14471-14484, Oct. 2021.
[4] M. Bastopcu and S. Ulukus, “Information Freshness in Cache Updating Systems,” IEEE Transactions on Wireless Communications, vol. 20, no. 3, pp. 1861-1874, March. 2021.
[5] S. Nath, J. Wu and J. Yang, “Delay and Energy Efficiency Tradeoff for Information Pushing System,” IEEE Transactions on Green Communications and Networking, vol. 2, no. 4, pp. 1027-1040, Dec. 2018.
[6] Y. Wen, W. Zhang and H. Luo, “Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones,” in 2012 Proceedings IEEE INFOCOM, pp. 2716-2720, 2012.
[7] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, “Prioritized experience replay,” in International Conference on Learning Representations, 2016.