MEC 환경에서 심층 강화학습을 이용한 오프로딩 기법의 성능비교

문성원, 임유진
숙명여자대학교 IT 공학과
sungwon268@sookmyung.ac.kr, yujin91@sookmyung.ac.kr

Performance Comparison of Deep Reinforcement Learning based Computation Offloading in MEC

Sungwon Moon, Yujin Lim
Dept. of IT Engineering, Sookmyung Women's University

요 약

5G 시대에 스마트 모바일 기기가 기하급수적으로 증가하면서 멀티 액세스 엣지 컴퓨팅(MEC)이 유망한 기술로 부상했다. 낮은 지연시간 안에 계산 집약적인 서비스를 제공하기 위해 MEC 서버로 오프로딩하는 특히, 태스크 도착률과 무선 채널의 상태가 확률적인 MEC 시스템 환경에서의 오프로딩 연구가 주목받고 있다. 본 논문에서는 차량의 전력과 지연시간을 최소화하기 위해 로컬 실행을 위한 연산 자원과 오프로딩을 위한 전송 전력을 할당하는 심층 강화학습 기반의 오프로딩 기법을 제안하였다. Deep Deterministic Policy Gradient (DDPG) 기반 기법과 Deep Q-network (DQN) 기반 기법을 차량의 전력 소비량과 큐잉 지연시간 측면에서 성능을 비교 분석하였다.

1. 서론

5G 시대에 스마트 모바일 기기가 기하급수적으로 증가하기 시작하면서, VR/AR, 얼굴 인식, 온라인 3D 게임 등 계산 집약적이고 지연시간에 민감한 모바일 어플리케이션들이 등장했다[1]. 하지만 이를 제한적인 컴퓨팅 자원과 배터리 용량을 가지는 무선 단말기에서 제공하기에는 한계가 있다. 그래서 이를 극복하기 위해 멀티 액세스 엣지 컴퓨팅(Multi-access Edge Computing, MEC)이 유망한 기술로 등장했다[2]. MEC는 클라우드와 같은 컴퓨팅 기능을 네트워크 엣지에 도입하여 단말기와 근접한 곳에서 고성능 서비스를 제공한다. 따라서 단말기는 계산 부하가 높고 시간이 걸리는 태스크를 중앙 집중형 클라우드 서버가 아닌 인근 MEC 서버(MECS)로 오프로딩하여 처리할 수 있다. 이를 통해 서비스의 지연시간과 단말기의 소비전력을 대폭 줄여 단말기의 QoE(Quality of Experience)를 향상시킬 수 있다.

이처럼 MEC 시스템은 큰 이점이 있지만 그에 반해 해결해야 할 과제가 있다. 실시간 어플리케이션들은 지연 제약조건이 엄격한데, 서비스 지연시간은 전송 및 전력 할당과 같이 오프로딩의 다양한 요소에 의해 영향을 받는다. 게다가 단말기는 전력도 한정되어 있고, 태스크가 동적으로 생성되므로 얼마만큼 오프로딩해야 하는지 결정하는 것은 어렵다. 특히, 무선 채널, 단말기의 위치 및 이동성이 확률적인 MEC 시스템에서 오프로딩을 결정하는 것은 어렵다.

MEC 시스템에서 태스크 오프로딩 및 자원 할당에 관한 연구들의 최적 목표는 대게 지연시간을 줄이거나[3,4], 에너지 소비를 줄이거나[5,6], 또는 에너지와 지연 시간 간의 균형을 맞추는 것[7,8]이다. 그러나 이런 연구들은 준정적 환경의 성능에 초점을 맞췄기 때문에 MEC 시스템의 동적인 측면을 고려하지 않았다. 따라서 동적인 측면을 고려하기 위해 강화학습 기반의 기법들이 제안되었다.

강화학습 기법 중 Q-learning 을 활용한 기법[1,9]이 제안되었으나, 단말기 수가 증가함에 따라 상태 공간의 크기가 커지면서 차원의 저주가 발생하여 효율성이 저하된다. 그래서 신경망을 적용한 심층 강화학습(Deep Reinforcement Learning, DRL)의 일종인 Deep Q-network (DQN) 기반의 기법들[2,10]이 제안되었다. 기존 기법들은 이산 행동 공간을 다루는 정책이기 때문에 에너지 또는 지연시간과 같은 연속적인 값을 다루는 데 한계가 있어 DRL 기법 중 연속 행동 공간을

다루는 Deep Deterministic Policy Gradient (DDPG) 기반의 기법들[1,11]이 제안되었다.

본 논문에서는 차량의 전력과 지연시간을 최소화하기 위해 DRL 기반의 오프로딩 기법을 제안하였으며, DDPG 기반 기법과 DQN 기반 기법을 전력과 지연시간 측면에서 성능을 비교 분석한다. 그리고 본 논문은 전기차와 같이 전력이 제한된 차량이 오프로딩 기기인 차량 엣지 컴퓨팅 시스템을 고려한다.

2. 시스템 모델

본 논문에서는 M개의 MECS 와 N개의 차량으로 구성된 MEC 시스템을 가정한다. 각 MECS 는 RSU (road side unit)에 배치되었으며, 차량의 위치와 가장 가까운 즉, 신호 세기가 가장 센 MECS m을 오프로딩 대상으로 선정하였다. MECS 는 충분히 높은 계산 능력을 갖추고 있어, MECS 로 오프로드된 모든 태스크들이 병렬적으로 처리된다고 가정한다. 그리고 각 차량들마다 $t \in T$에 평균 태스크 도착률 λ를 바탕으로 $z_n(t)$만큼의 태스크가 발생하고, 태스크는 파티셔닝이 가능하여 태스크의 일부가 MECS 로 오프로딩 할 수 있다.

$H_{n,m}(t)$는 차량 n과 MECS m사이의 채널 이득이고, σ^2은 잡음 전력, B는 전송 대역폭이다. $H_{n,m}(t)$과 차량의 송신 전력 $p_n^{tr}(t)$에 따라 차량 n이 MECS m으로 오프로딩할 수 있는 데이터의 양이다.

$$d_n^{tr}(t) = Blog_2\left(1 + \frac{p_n^{tr}(t)H_{n,m}(t)}{\sigma^2}\right) \quad (1)$$

$f_n(t) \in [0, F_n]$는 차량 n에서 로컬 실행을 위해 할당된 CPU 주파수이고, C_n는 차량 n에서 태스크 한 비트 당 요구하는 CPU 사이클 수이다. 차량 n에서 로컬로 수행할 수 있는 데이터의 양이다.

$$d_n^{lo}(t) = \frac{f_n(t)}{C_n} \quad (2)$$

이 데이터의 양을 수행할 때 소비되는 전력은 다음과 같고, η는 칩 아키텍처와 관련된 에너지 계수이다.

$$p_n^{lo}(t) = \eta\left(f_n(t)\right)^3 \quad (3)$$

t 에 $z_n(t)$만큼의 태스크가 발생하고, (1)와 (2)만큼 처리되면, 차량 n의 큐의 길이는 다음과 같다.

$$q_n(t+1) = \left[q_n(t) - \left(d_n^{lo}(t) + d_n^{tr}(t)\right)\right]^+ + z_n(t) \quad (4)$$

본 기법은 차량의 전력 소모량은 줄이면서, 태스크의 처리량은 높여 큐잉 지연시간을 최소화하는 것이 목표이다. 리틀의 법칙(Little's Theorem)에 따라 태스크의 평균 큐 길이는 큐잉 지연시간과 비례한다. 따라서 $\omega \in [0,1]$를 사용하여 가중치 합계로 정의하고, 제약조건들을 고려하여 식으로 표현하면 다음과 같다.

$$\min \quad \omega\left(p_n^{lo}(t) + p_n^{tr}(t)\right) + (1-\omega)q_n(t), \ \forall t \in T \quad (5)$$

s.t

$$f_n(t) \in [0, F_n], \forall t \in T \quad (6)$$

$$p_n^{tr}(t) \in [0, P_n^{tr}], \forall t \in T \quad (7)$$

3. 제안하는 알고리즘

본 기법은 MECS 로 최적의 오프로딩을 하기 위해 로컬 실행을 위한 최적의 연산 자원과 오프로딩을 위한 최적의 전송 전력을 할당함으로써 차량의 전력 소모량과 큐잉 지연시간을 최소화하는 DRL 기반의 오프로딩 기법을 제안하였다.

DQN 은 Q-learning 의 Q 테이블을 신경망으로 치환하여 학습하는 알고리즘이다. 신경망을 이용하여 Q 함수를 근사할 수 있도록 학습시키며, 이를 $Q^*(s,a|\theta)$로 표현하고, θ 는 메인 네트워크의 파라미터이다. DQN 네트워크는 타깃 네트워크$Q(s',a'|\theta')$와 메인 네트워크 $Q(s,a|\theta)$의 MSE(mean square error)를 줄여나가는 방향으로 다음과 같이 학습한다.

$$L(\theta) = \mathbb{E}\left[\left(r + \gamma \max_{a'} Q(s',a'|\theta') - Q(s,a|\theta)\right)^2\right] \quad (8)$$

그리고 ϵ-탐욕적 기법을 사용하여 가끔 보상에 따른 탐색이 아닌 무작위성으로 탐사하는 방법을 사용한다.

DDPG 는 DQN 과 액터-크리틱 (actor-critic)을 결합한 알고리즘이다. 정책 함수를 근사하는 액터 네트워크 $\mu(s|\theta^\mu)$ 와 Q 함수를 근사하는 크리틱 네트워크 $Q(s,a|\theta^Q)$를 학습한다. 그리고 DQN 과 동일하게, 타깃 네트워크를 활용하여 액터와 크리틱 네트워크를 복제하여 액터 및 크리틱 타깃 네트워크 $\mu'(s|\theta^{\mu'})$, $Q'(s,a|\theta^{Q'})$를 생성하여 목표 값을 계산한다. θ^μ, $\theta^{\mu'}$, θ^Q, $\theta^{Q'}$은 네 개의 신경망의 파라미터이다.

액터 네트워크는 정책 경사(policy gradient) 방식으로 다음과 같이 학습한다.

$$\nabla_{\theta^\mu}\mathcal{J} \approx \mathbb{E}[\nabla_a Q(s,a|\theta^Q)\nabla_{\theta^\mu}\mu(s|\theta^\mu)] \quad (9)$$

크리틱 네트워크는 DQN 과 비슷하여 식 (8)처럼 다음과 같이 손실함수를 최소화하는 방향으로 학습한다.

$$L(\theta^Q) = \mathbb{E}[\left(r + \gamma(Q(s,\mu'(s|\theta^{\mu'})|\theta^{Q'}) - Q(s,a|\theta^Q)\right)^2] \quad (10)$$

Off-policy 기법으로써 DDPG 는 탐험과 학습 알고리즘을 독립적으로 다룰 수 있어, 탐험 정책 μ'은 액터 정책에 노이즈 $\Delta\mu$을 다음과 같이 추가한다.

$$\mu'(s) = \mu(s|\theta^\mu) + \Delta\mu \quad (10)$$

심층 신경망을 학습시키기 용이하게 하기 위해 데이터 간의 상관관계를 최소화하는 리플레이 버퍼 (replay buffer)를 사용한다. 그리고 타깃 네트워크들을 업데이트할 때, 다음과 같이 완화적으로 진행하여 학습과정에서 타깃 값이 빠르게 변화하는 것을 억제하여 학습의 안정성을 향상시킨다.

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'} \quad (11)$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'} \quad (12)$$

DQN 과 DDPG 을 이용한 오프로딩 기법은 각각 표 1 과 표 2 와 같다. 각 차량들이 에이전트로 오프로딩 정책을 독립적으로 학습하며, 전력과 지연시간을 최소화하는 최적의 정책을 찾는다. 이를 MDP(markov decision process)로 정의하면 다음과 같다.

차량 에이전트들은 최적의 오프로딩 정책을 결정하기 위해 t마다 차량의 큐의 길이 $q_n(t)$와 무선채널 상태 정보 $H_{n,m}(t)$을 관찰한다. 따라서 t에서 에이전트의 상태는 $s_n(t) = (q_n(t),\ H_{n,m}(t))$로 정의한다.

MEC 시스템에서 차량 에이전트는 MECS 로 오프로딩하기 위해 최적의 로컬 실행 자원과 최적의 오프로

드 전력을 할당하는 최적의 오프로드 전략을 찾아야 한다. 따라서 t에서 에이전트의 행동은 $a_n(t) = \{f_n(t), p_n^{tr}(t)\}$으로 정의한다. 여기서, $f_n(t)$는 로컬 실행을 위해 할당된 CPU 주파수이고, $p_n^{tr}(t)$는 오프로딩 실행을 위한 전송 전력이다.

에이전트는 행동을 취함으로써 보상을 얻는데, 이때 가장 높은 보상을 줄 행동을 선택한다. 보상 함수는 목적 함수와 관련이 있는데, 본 논문의 목표는 차량의 전력과 지연시간을 최소화하는 것으로 보상과 목적에 음의 상관관계가 있다. 따라서 에이전트의 보상은 식 (5)을 이용하고 부정적인 보상으로 정의하여 $r_n(t) = -\omega\left(p_n^{lo}(t) + p_n^{tr}(t)\right) - (1-\omega)q_n(t)$이다.

<표 1> DQN 기반 오프로딩 기법

Algorithm 1. DQN based Computation Offloading

Initialize main network $Q(s,a|\theta)$
Initialize target network $Q(s,a|\theta')$with weights $\theta' = \theta$
Initialize the experience replay buffer
for each episode **do**
 Initialize environment and state $s_n(t)$ for each agent $n \in N$
 for each time slot $t \in T$ **do**
  for each agent $n \in N$ **do**
   if random number $\leq \epsilon$:
    Select action $a_n(t)$ randomly the CPU frequency and power from the discrete action space in the MDP
   else:
    Select action $a_n(t) = \max_a Q(s,a)$
   Execute the action $a_n(t)$ and receive reward $r_n(t)$
   Store $(s_n(t), a_n(t), r_n(t), s_n(t+1))$ in replay buffer
   Randomly sample a mini-batch of samples
   Perform a gradient descent step via (8)
   Update $\theta' = \theta$

<표 2> DDPG 기반 오프로딩 기법

Algorithm 1. DDPG based Computation Offloading

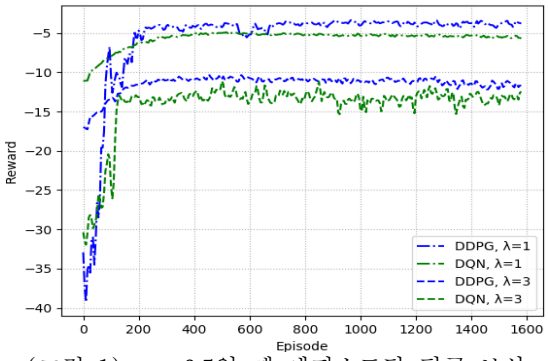
Initialize critic network $Q(s,a|\theta^Q)$ and actor network $\mu(s|\theta^\mu)$
Initialize target network with weights $\theta^{Q\prime} = \theta^Q$ and $\theta^{\mu\prime} = \theta^\mu$
Initialize the experience replay buffer
for each episode **do**
 Initialize environment and state $s_n(t)$ for each agent $n \in N$
 for each time slot $t \in T$ **do**
  for each agent $n \in N$ **do**
   Determine the CPU frequency and power by selecting an action $a_n(t) = \mu(s|\theta^\mu) + \Delta\mu$
   Execute the action $a_n(t)$ and receive reward $r_n(t)$
   Store $(s_n(t), a_n(t), r_n(t), s_n(t+1))$ in replay buffer
   Randomly sample a mini-batch of samples
   Update the critic network and the actor network via (10) and (9), respectively
   Update the target networks via (11) and (12)

4. 실험 및 성능 비교 분석

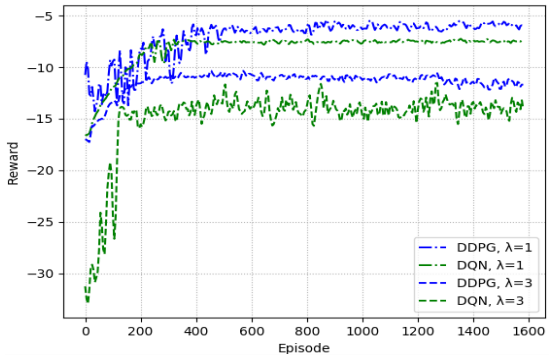
제안한 오프로딩 기법에 대한 성능을 측정하기 위해 실험을 진행했다. 총 3 개의 MECS 가 배치된 환경이며, MECS 의 통신반경은 $250m$이다. 각 차량은 태스크 도착률 λ $Mbps$ 기반으로 각 타임슬롯 t마다 태스크가 발생하며, 타임슬롯의 길이는 $1ms$이다. 실험에서 전송 대역폭 B 는 $10MHZ$, 잡음 전력 σ^2 은 $10^{-9}W$으로 설정했다. 그리고 차량의 최대 전송 전력 P_n^{tr}은 2W, 최대 CPU 주기 F_n는 $1GHZ$, 데이트 한 비트 당 요구되는 CPU 사이클 C_n는 $500cycles/bit$이고, 에너지 계수 η 는 10^{-27}이다.

DQN 은 1 개의 히든 레이어가 있는 3 계층 완전 연결 신경망으로, DDPG 의 액터와 크리틱 네트워크는 각 2 개의 히든 레이어가 있는 4 층 완전 연결 신경망으로 구성하였고, 활성화 함수는 ReLU 를 사용했다. DQN 의 학습률은 1e-4, DDPG 에서 액터와 크리틱 네트워크의 학습률은 각각 1e-5 와 1e-4 이다. 리플레이 버퍼의 크기는 2.5×10^5이며, 배치 크기는 128 이다. 그리고 γ는 0.99, ϵ는 1.0, τ는 0.01 이다.

본 논문에서는 DRL 의 DDPG 와 DQN 을 적용한 오프로딩 기법 간 성능을 비교하고자 한다. DDPG 와 달리, DQN 은 연속 행동 공간을 지원하지 않는다. 그래서 DQN 은 연속적인 문제에 대해 $l-$레벨 행동 공간을 고안하여, 사용 가능한 CPU 주파수 집합을 $F = \{0, \frac{F_n}{l}, \frac{2F_n}{l}, \ldots, F_n\}$으로 정의하고, 사용 가능한 전송 전력의 집합을 $P = \{0, \frac{P_n^{tr}}{l}, \frac{2P_n^{tr}}{l}, \ldots, P_n^{tr}\}$으로 정의한다. 따라서 각 에이전트의 행동 공간은 두 집합 F와 P의 데카르트 곱으로 $F \times P$ 이며, $l = 8$로 설정하였다.



(그림 1) $\omega = 0.5$일 때 에피소드당 평균 보상



(그림 2) $\omega = 0.7$일 때 에피소드당 평균 보상

그림 1 과 그림 2 는 각각 ω를 0.5 와 0.7 로 설정하여 에피소드당 평균 보상을 태스크 도착률 λ가 $1Mbps$와 $3Mbps$ 일 때 두 기법 간 성능을 비교하였다. DDPG 와 DQN 기반 기법 모두 학습 단계가 증가할수록 즉, 차량 에이전트와 MEC 시스템 환경 간 상호 작용이 지속될수록 각 에피소드당 평균 보상이 증가함을 관찰할 수 있다. 이는 오프로딩 기법이 성공적으로 학습됨을 나타내며, 약 1200 번 수행 이후에는 학습이 안정된다. DDPG 기반 오프로딩 기법이 DQN 기반 오프로딩 기법의 성능보다 약 22% 우수하다. 본 논문은 자원과 전력 같은 연속적인 값을 다루기 때문에 연속 행동 공간 기반 DDPG 기법의 성능이 이산 행동 공간 기반 DQN 기법에 비해 좋게 나온다. 특히, DQN 은 연속 행동 공간을 다루지 않고 이산 행동 공간만 다루기 때문에 연속적인 문제에서는 $l-$레벨 행동 공간을 이용한다. 결과적으로, 연속적인 문제에 대해 DDPG 기법이 DQN 기법보다 더 효율적으로 행동 공간을 탐색할 수 있음을 확인할 수 있다.

그리고 태스크 도착률이 증가할수록 에피소드당 평균 보상이 낮아지는 것을 관찰할 수 있는데, 이는 태스크의 연산 수요가 많아질수록 더 많은 전력을 소비하고 더 긴 큐잉 지연시간이 발생하기 때문이다. 그리고 ω=0.7 인 그림 1 이 ω=0.5 인 그림 2 에서의 두 기법 사이의 성능 격차보다 약간 크다. ω가 커질수록 전력 소비에 더 많은 비중을 둬 지연시간을 희생하여 전력 소비량을 감소시킨다. 그러면 각 차량의 소비 전력은 낮아지고, 낮아진 전력만큼 태스크의 처리량이 감소하여 큐잉 지연 시간이 증가한다.

5. 결론

본 논문에서는 무선 채널, 단말기의 위치 및 이동성이 확률적인 차량 엣지 컴퓨팅 환경에서 차량의 전력과 큐잉 지연시간을 최소화하기 위해 DRL 기반의 오프로딩 기법을 제안하였다. 각 차량이 오프로딩 정책을 독립적으로 학습하여, 로컬 실행을 위한 자원과 오프로딩을 위한 전송 전력을 할당한다. 연속 행동 기반 DDPG 기법과 이산 행동 공간 기반 DQN 기법을 비교하기 위해 실험을 진행하였다. 비교 결과, DDPG 기법이 DQN 기법에 비해 연속적인 문제에 대해 더 효율적임을 확인하였다. 향후 MECS 의 큐도 고려하여 오프로딩 전략을 결정하고, 각 에이전트가 독립적으로 학습하는 것이 아닌 서로 협력하여 전체적인 보상에 대해 어느 정도 공헌을 하는지 판단할 수 있는 Counterfactual multi-agent (COMA) 강화학습 기반의 오프로딩 연구를 진행하고자 한다.

사사문구

이 성과는 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. 2021R1F1A1047113).

참고문헌

[1] Z. Cheng, M. Min, M. Liwang, L. Huang and Z. Gao, "Multiagent DDPG-Based Joint Task Partitioning and Power Control in Fog Computing Networks," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 104-116, Jan. 2022.

[2] M. Tang and V. W. S. Wong, "Deep Reinforcement Learning for Task Offloading in Mobile Edge Computing Systems," *IEEE Transactions on Mobile Computing (Early Access),* Nov. 2020.

[3] Y. Mao, J. Zhang and K. B. Letaief, "Dynamic Computation Offloading for Mobile-Edge Computing with Energy Harvesting Devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590-3605, Dec. 2016.

[4] J. Zhang, H. Guo, J. Liu and Y. Zhang, "Task Offloading in Vehicular Edge Computing Networks: A Load-Balancing Solution," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 2092-2104, Feb. 2020.

[5] H. Guo, J. Liu and J. Zhang, "Efficient Computation Offloading for Multi-Access Edge Computing in 5G HetNets," *2018 IEEE International Conference on Communications (ICC)*, Kansas City, USA, 2018, pp. 1-6.

[6] O. Muñoz, A. Pascual-Iserte and J. Vidal, "Optimization of Radio and Computational Resources for Energy Efficiency in Latency-Constrained Application Offloading," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 10, pp. 4738-4755, Oct. 2015.

[7] M. Chen and Y. Hao, "Task Offloading for Mobile Edge Computing in Software Defined Ultra-Dense Network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587-597, Mar. 2018.

[8] J. Zhang, X. Hu, Z. Ning, E. Ngai, L. Zhou, J. Wei, J. Cheng and B. Hu, "Energy-Latency Tradeoff for Energy-Aware Offloading in Mobile Edge Computing Networks," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2633-2645, Aug. 2018.

[9] F. Jiang, W. Liu, J. Wang and X. Liu, "Q-Learning based Task Offloading and Resource Allocation Scheme for Internet of Vehicles," *2020 IEEE/CIC International Conference on Communications in China (ICCC)*, Chongqing, China, 2020, pp. 460-465.

[10] Y. Liu, H. Yu, S. Xie and Y. Zhang, "Deep Reinforcement Learning for Offloading and Resource Allocation in Vehicle Edge Computing and Networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 11, pp. 11158-11168, Nov. 2019.

[11] H. Wang, H. Ke, G. Liu and W. Sun, "Computation Migration and Resource Allocation in Heterogeneous Vehicular Networks: A Deep Reinforcement Learning Approach," *IEEE Access*, vol. 8, pp. 171140-171153, Sep. 2020.