Machine Learning for Caching Placement in Edge Computing Networks

Liang Zhang, Bijan Jabbari

lzhang36@gmu.edu

Communications and Networks Laboratory,

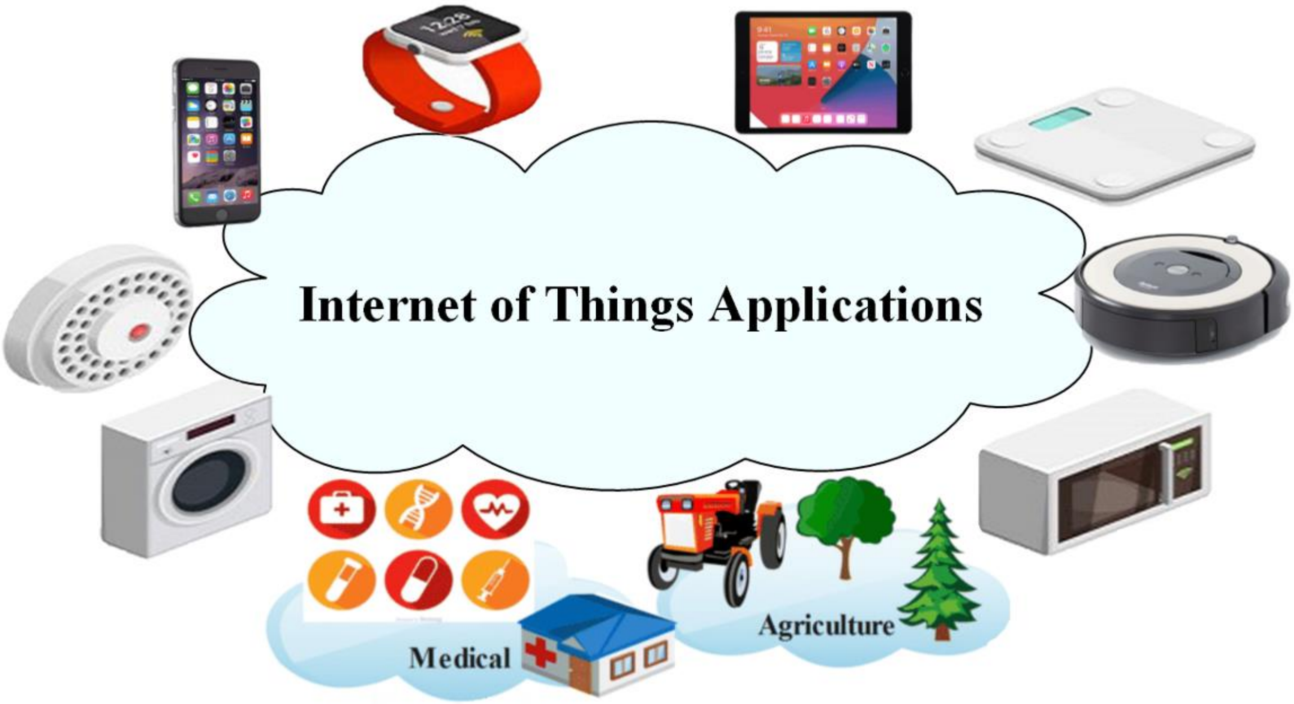
Department of ECE, George Mason University, Fairfax, VA 22030 USA

Outline

- Caching Placement in the Edge Computing Network
- System Model and Problem Formulation
- Algorithm and Analysis
- Evaluation Results
- Conclusions

Internet of Things Applications

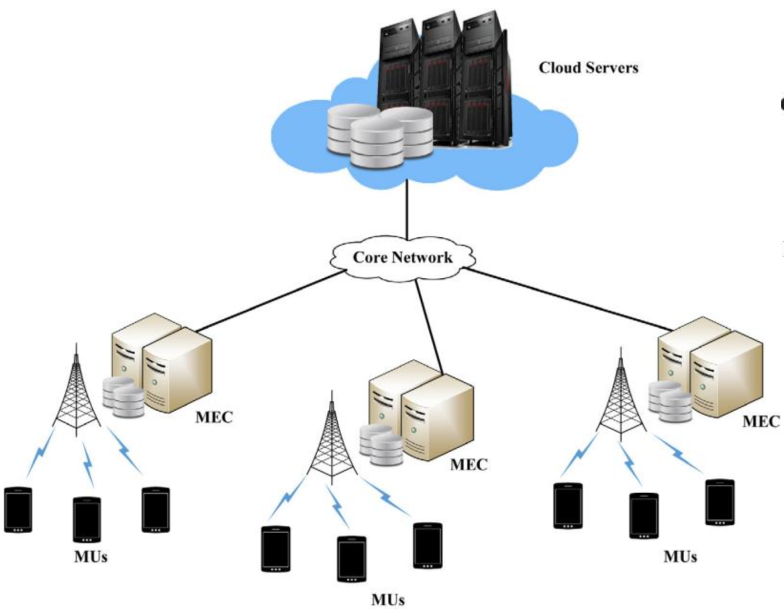
- Billions of Internet of Things (IoT) devices are connected to the Internet, and many IoT devices have limited power, computing and storage resources [1].
- Various IoT applications, have been widely studied to improve our daily life. Different applications may have various resource preference.



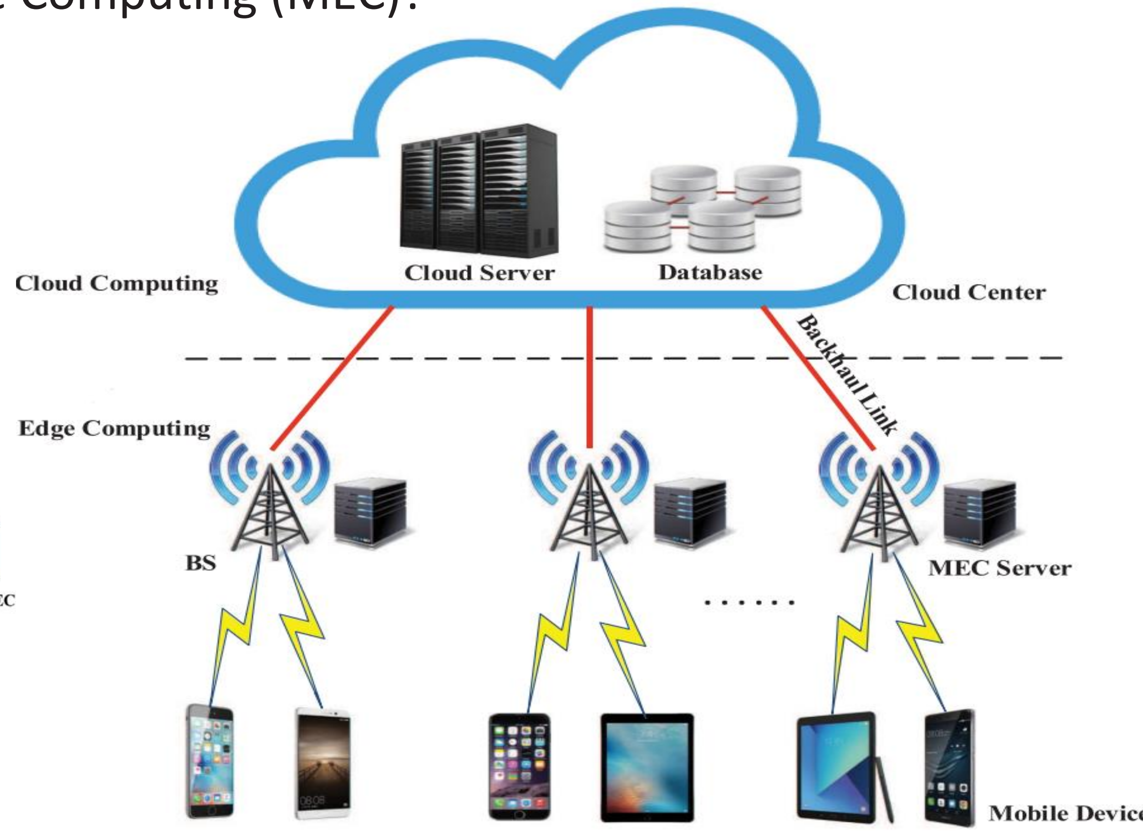
Source: C. Qiu et al., "Networking integrated cloud–edge–end in IoT: A blockchain–assisted collective Q-learning approach," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12 694–12 704, Aug. 2021.

Mobile Edge Computing

- MEC is a network architecture that pushes cloud computing capabilities at edge nodes that are close to users and connected to cloud servers via a core network.
- Why do we need Mobile Edge Computing (MEC)?



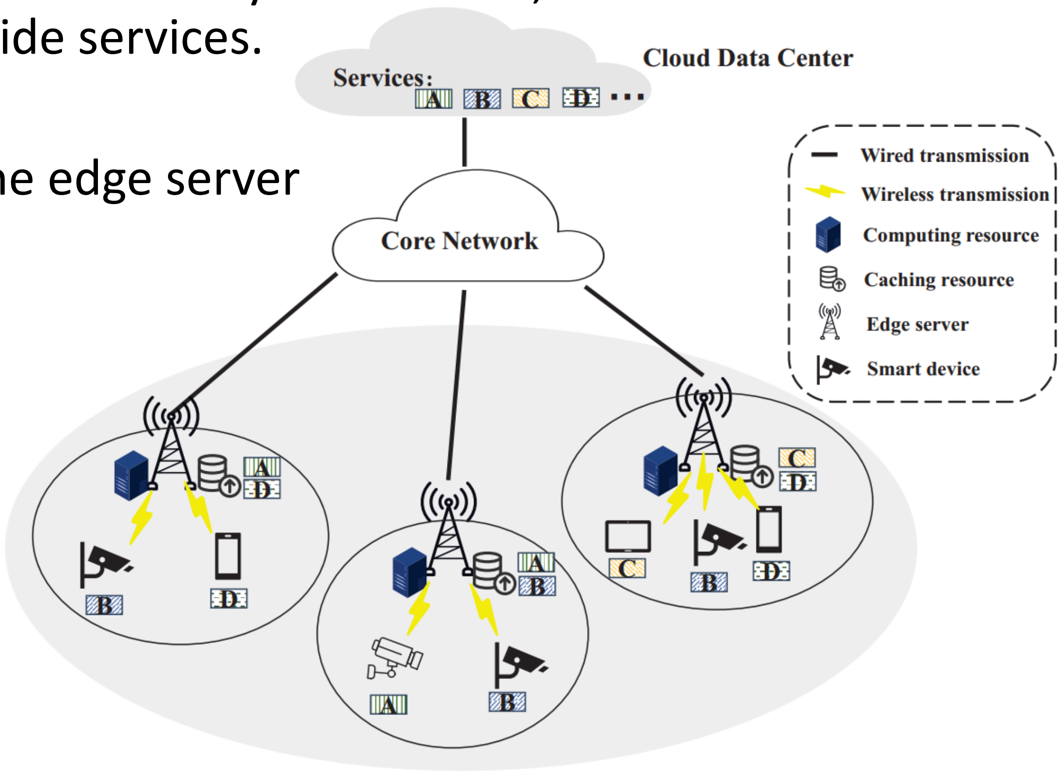
Source: I.A. Elgendy, et al. / Future Generation Computer Systems, 100 (2019) 531–541.



Source: J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 5031–5044, May 2019.

Service Caching in the Edge Computing Networks

- Service caching means that the edge servers cache application services and related databases, so that they can process corresponding computing tasks to reduce completion time and energy consumption.
- Taking intelligent monitoring in smart grid as an example: processing computing tasks requires not only sensor data, but also data from surrounding infrastructure to provide services.
- Data from the IoTD
- Background data in the edge server



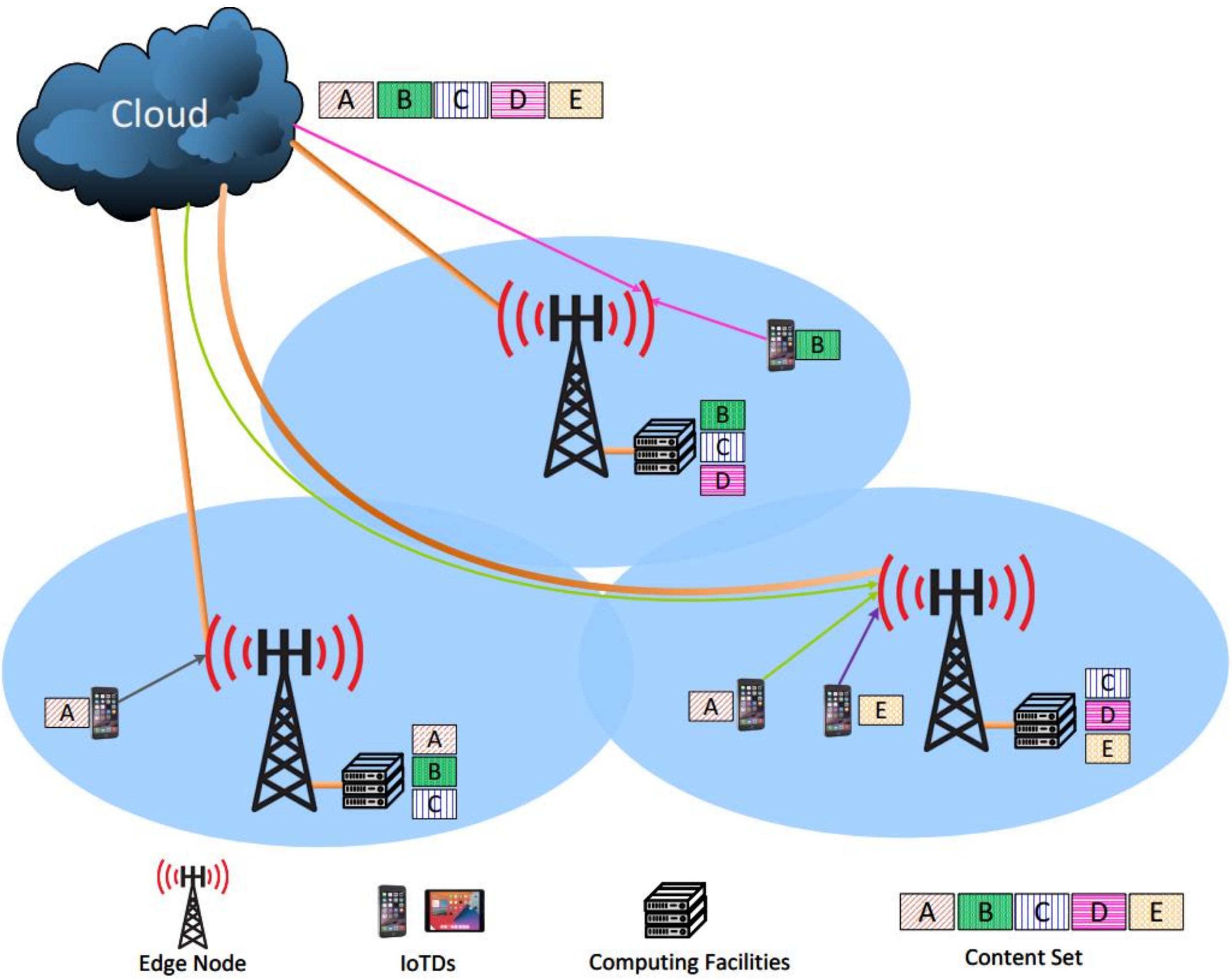
Source: H. Zhou, Z. Zhang, D. Li, and Z. Su, "Joint optimization of computing offloading and service caching in edge computing-based smart grid," *IEEE Trans. Cloud Comput.*, pp. 1–1, 2022.

Outline

- Caching Placement in the Edge Computing Network
- System Model and Problem Formulation
- Algorithm and Analysis
- Evaluation Results
- Conclusions

Multi-content Caching in the Edge Computing Network

- We studied the multi-content placement (*MCP*) problem in edge-computing networks for IoTDs.



System Model

- We assume the popularity of the contents of IoTDs follows the Zipf distribution, and the probability of requesting content k by IoTD i is:

$$q_i^k = \frac{\xi^{-k}}{\sum_1^K \xi^{-k}}\,, \tag{1}$$

- Denote α_j^k as the indicator of the caching status of content *k* in edge node *j*; it is a binary variable and equals to 1 if content *k* is cached by edge node *j*; otherwise, it is 0.

- Let $t_{i,j}$, $t_{i,j}^1$ and $t_{i,j}^2$ be the total service delay, and the delay with the content cached ($\alpha_j^k = 1$) and the delay without the content cached ($\alpha_j^k = 0$) :

$$t_{i,j} = \alpha_j^k t_{i,j}^1 + (1 - \alpha_j^k) t_{i,j}^2, \quad \forall i \in \mathscr{U}\,, \tag{4}$$

$$t_{i,j}^1 = \frac{r_i}{d_{i,j}} + \frac{c_i}{\zeta_{i,j}}, \quad \forall i \in \mathscr{U}\,, \tag{5}$$

$$t_{i,j}^2 = \frac{r_i}{d_{i,j}} + \frac{c_i}{\zeta_{i,j}} + t_{i,j}^{cloud}, \quad \forall i \in \mathscr{U}\,, \tag{6}$$

Problem Formulation

- We formulate the multi-content placement (MCP) problem in edge-computing networks for IoTDs with the target of minimizing the average latency of IoTDs.

$$
\begin{aligned}
&\mathscr{P}_0: \min_{\alpha_j^k, \omega_{i,j}, \beta_{i,j}, \zeta_{i,j}} \quad \frac{1}{|\mathscr{U}|} \sum_i \sum_j t_{i,j} \\
&s.t.: \\
&\qquad C1: \sum_k \alpha_j^k \leq \Gamma, \quad \forall j \in \mathscr{E}, \\
&\qquad C2: \sum_j \omega_{i,j} \leq 1, \quad \forall i \in \mathscr{U}, \\
&\qquad C3: \sum_i \omega_{i,j} \beta_{i,j} \leq f_j, \quad \forall j \in \mathscr{E}, \\
&\qquad C4: \sum_i \omega_{i,j} \zeta_{i,j} \leq C_j, \quad \forall j \in \mathscr{E}, \\
&\qquad C5: \alpha_j^k \in \{0,1\}, \quad \forall j \in \mathscr{E}, k \in \mathscr{K}. \\
&\qquad C6: \omega_{i,j} \in \{0,1\}, \quad \forall i \in \mathscr{U}, j \in \mathscr{E}. \qquad\qquad (7)
\end{aligned}
$$

Outline

- Caching Placement in the Edge Computing Network
- System Model and Problem Formulation
- Algorithm and Analysis
- Evaluation Results
- Conclusions

Problem Analysis

- The MCP problem is NP-hard because it is a non-convex, nonlinear, and mixed discrete optimization problem [11], [12].

- Then, a machine learning algorithm is proposed to solve the MCP problem. First, we focus on the resource assignment based on the given caching placement and IoTD assignment. Second, we employ a machine learning algorithm to obtain the best result of the caching placement and the IoTD assignment, and then the MCP problem is solved.

Problem Analysis (Cont'd)

- For a given caching status and service status, the MCP problem can be reformulated as follows:

$$\mathscr{P}_1: \min_{\beta_{i,j},\zeta_{i,j}} \quad \frac{1}{|\mathscr{U}|}\sum_i \sum_j t_{i,j}$$
$$s.t.:$$
$$C1: \sum_i \omega_{i,j}\beta_{i,j} \le f_j, \quad \forall j \in \mathscr{E},$$
$$C2: \sum_i \omega_{i,j}\zeta_{i,j} \le C_j, \quad \forall j \in \mathscr{E}. \qquad (8)$$

$$\mathscr{P}_2: \min_{\beta_{i,j},\zeta_{i,j}} \quad \frac{1}{|\mathscr{U}|}\sum_i \sum_j t_{i,j}$$
$$s.t.:$$
$$C1: \sum_i \omega_{i,j}\beta_{i,j} \le f_j, \quad \forall j \in \mathscr{E}. \qquad (9)$$

Optimal Resource Assignment

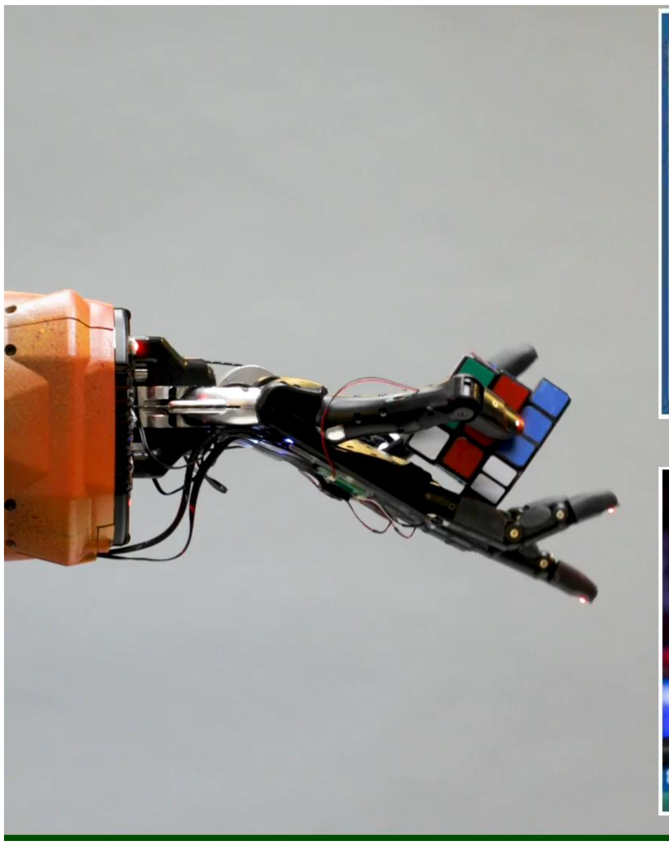
Algorithm 1: Optimal Resource Assignment

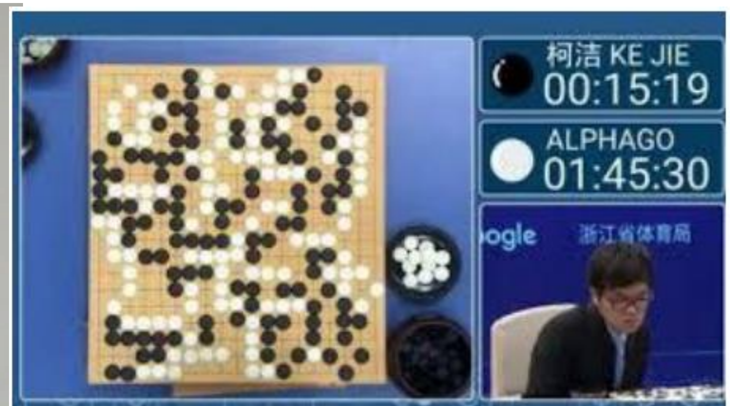
Input : $\mathscr{B}$, $\mathscr{U}$, f_j, C_j, α_j^k and $\omega_{i,j}$;
Output: $\beta_{i,j}$ and $\zeta_{i,j}$;

for j *in* $\mathscr{B}$ **do**
    determine $\mathscr{U}_j = \{\omega_{i,j} = 1\}$;
    initialize $t_{i,j}$, $\Delta t_{i,j}$, β_0 and ζ_0;
    set $f_j^1 = f_j$, $C_j^1 = C_j$, $f_j^2 = 0$, and $C_j^2 = 0$;
    calculate $t_{i,j}^{total} = \sum_i \sum_j t_{i,j}$;
    while $f_j^1 \geq \Delta b \,\&\, C_j^1 \geq \Delta\tau$ **do**
        for i *in* $\mathscr{U}_j$ **do**
            update $t_{i,j}$ by $t_{i,j}^{'}$ if β_0 and ζ_0 are assigned to this IoTD;
            calculate the latency reduction $\Delta t_{i,j} = t_{i,j} - t_{i,j}^{'}$;
        find $(i', j') = \underset{i}{\operatorname{argmax}}\, \Delta t_{i,j}$;
        assign β_0 and ζ_0 to IoTD i' by edge node j';
        $\beta_{i',j'} = \beta_{i',j'} + \beta_0$;
        $\zeta_{i',j'} = \zeta_{i',j'} + \zeta_0$;
        update $t_{i,j}^{total}$;
update $\beta_{i,j}$ and $\zeta_{i,j}$;

What is Machine Learning?

- Machine learning is a branch of artificial intelligence and computer science, which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy*.

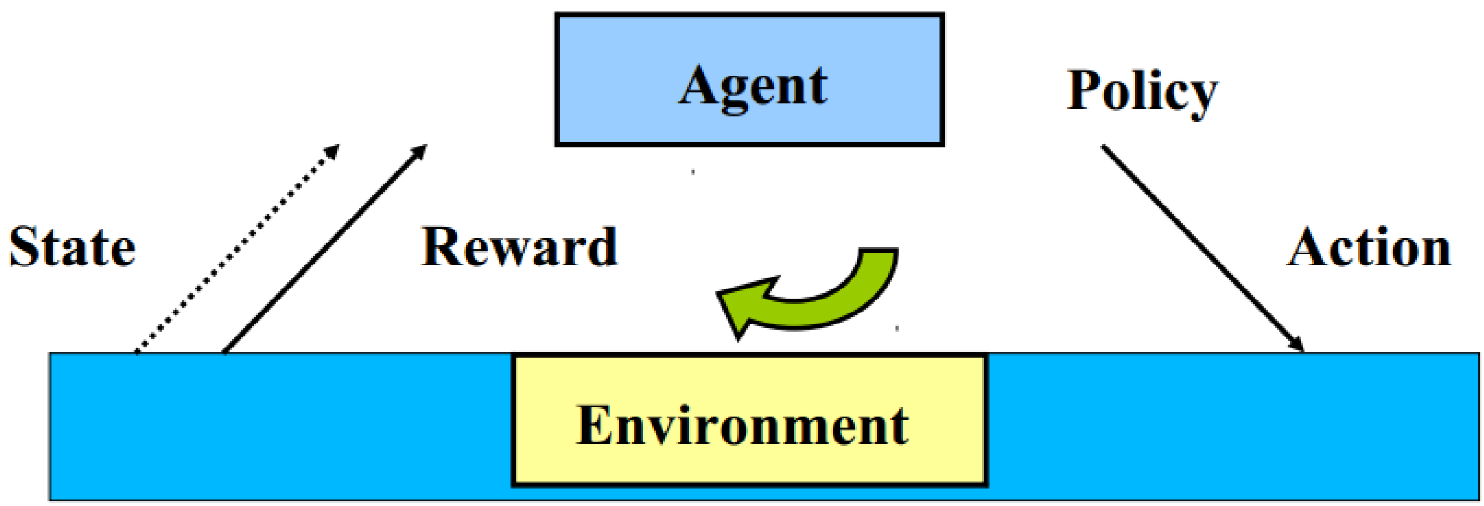




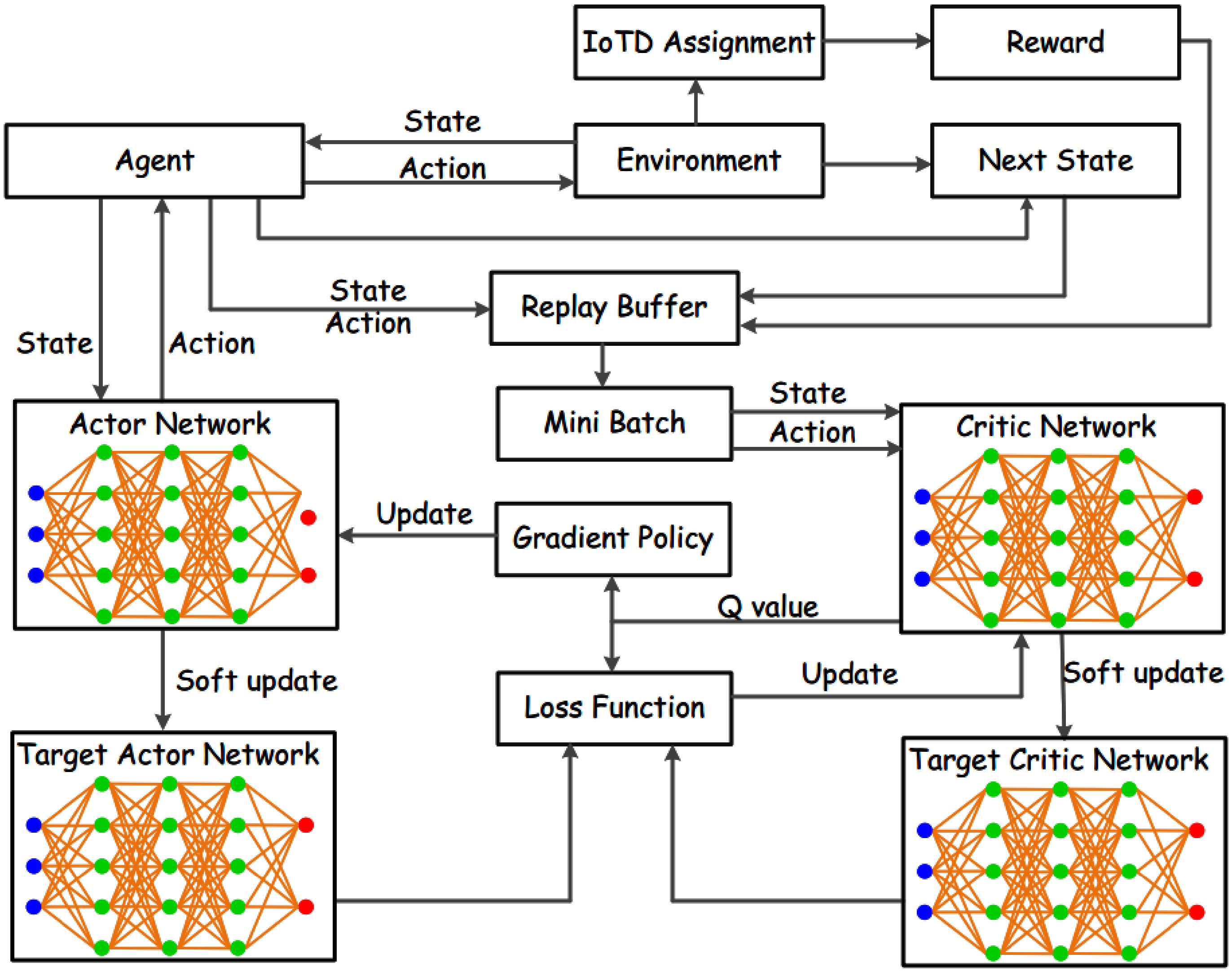
 [*] Source: https://www.ibm.com/cloud/learn/machine-learning

Elements of Reinforcement Learning

- Agent: Intelligent programs
- Environment: External condition
- A typical fully connected neural network includes three layers: the input layer, the hidden layer and the output layer.
- The output (action $a(t)$) is determined by the input (state $s(t)$), the reward ($r(t)$) is determined by the output, and the weights are updated based on the reward.



The Diagram of the DDPG-MCP Algorithm



The framework of DDPG

- State: $s(n) = \{XY(B), XY(U), c_i\ , e_i, r_i\}$
- Action: $a(n) = \{\alpha_{i,j}, \omega_{i,j}\}$
- Reward: $g(n) = -\frac{1}{|U|}\sum_i \sum_j t_{i,j}$

- **Actor network**: input is s, output is a
- **Critic network**: input is (s, a), output is Q(s,a)
- **Target actor network and target critic network:**

  input is actor network and critic network

  it is used to calculate the loss function of the critic network

The framework of DDPG

- The critic network is updated based on the loss function:

$$L(\theta^Q) = \frac{1}{\Gamma}\sum_{\gamma=1}^{\Gamma}[g(\gamma) - Q(s(\gamma), a(\gamma)|\theta^Q) + \lambda A_1]^2. \quad (10)$$

- The actor network is updated based on the gradient policy as follows.

$$\nabla_{\theta^\phi}\Omega(\theta^\phi) = \frac{1}{\Gamma}\sum_{\gamma=1}^{\Gamma}(A_2 \nabla_a Q(s, a|\theta^Q)\big|_{s=s(\gamma), a=\phi(s(\gamma))}).$$

- Then, the target networks are updated as:

$$\theta^{\phi-} \leftarrow \varsigma\theta^\phi + (1-\varsigma)\theta^{\phi-}$$
$$\theta^{Q-} \leftarrow \varsigma\theta^Q + (1-\varsigma)\theta^{Q-}$$

The DDPG-MCP Algorithm

Algorithm 2: DDPG-MCP

Input : $\mathscr{B}$, $\mathscr{U}$ and four neural networks;
Output: $g(n)$, α_j^k and $\omega_{i,j}$;

for *epoch* m **do**
  Initialize the actor network with the weight θ^{ϕ};
  set the target actor network with the weight $\theta^{\phi-}$;
  Initialize the critic network with the weight θ^{Q};
  set the target critic network with the weight θ^{Q-};
  set the replay buffer $\eta = 0$ and initialize η^b;
  $s(n) = 0$, $a(n) = 0$ and $g(n) = 0$;
  for *each training step* n **do**
    calculate state $s(n+1)$;
    add $\{s(n), a(n), g(n), s(n+1)\}$ to replay buffer;
    $\eta = \eta + 1$;
    if $\eta \geq \eta^b$ **then**
      update the critic network θ^Q by Eq. (10);
      update actor network θ^{ϕ} by Eq. (11);
      update target networks;
    get $a(n+1)$;
    add noise: $a(n+1) = a(n+1) + \sigma(n+1)$;
    decode $a(n+1)$ to obtain α_j^k and $\omega_{i,j}$ by Algorithm 1;
    calculate $g(n+1)$;
find the largest reward $g(n')$;
obtain α_j^k and $\omega_{i,j}$ by $(\alpha_j^k, \omega_{i,j}) = \operatorname*{argmax}_{i,j,k} \cup g(n)$;
return $g(n)$, α_j^k and $\omega_{i,j}$.

Update Critic network

Update Actor network

Update Target Actor network

Update Target Critic network

Outline

- Caching Placement in the Edge Computing Network
- System Model and Problem Formulation
- Algorithm and Analysis
- Evaluation Results
- Conclusions

Simulation Settings

- We use Python3.8 and Tensorflow 2.6.2 (tf.keras.optimizers.Adam) to run our simulations.
- Three baseline algorithms are utilized to evaluate the performance of the DDPG-RATE algorithm: (1) Fixed-Best-MCP, (2) Fixed-Fair-MCP and (3) Random-Fair-MCP.

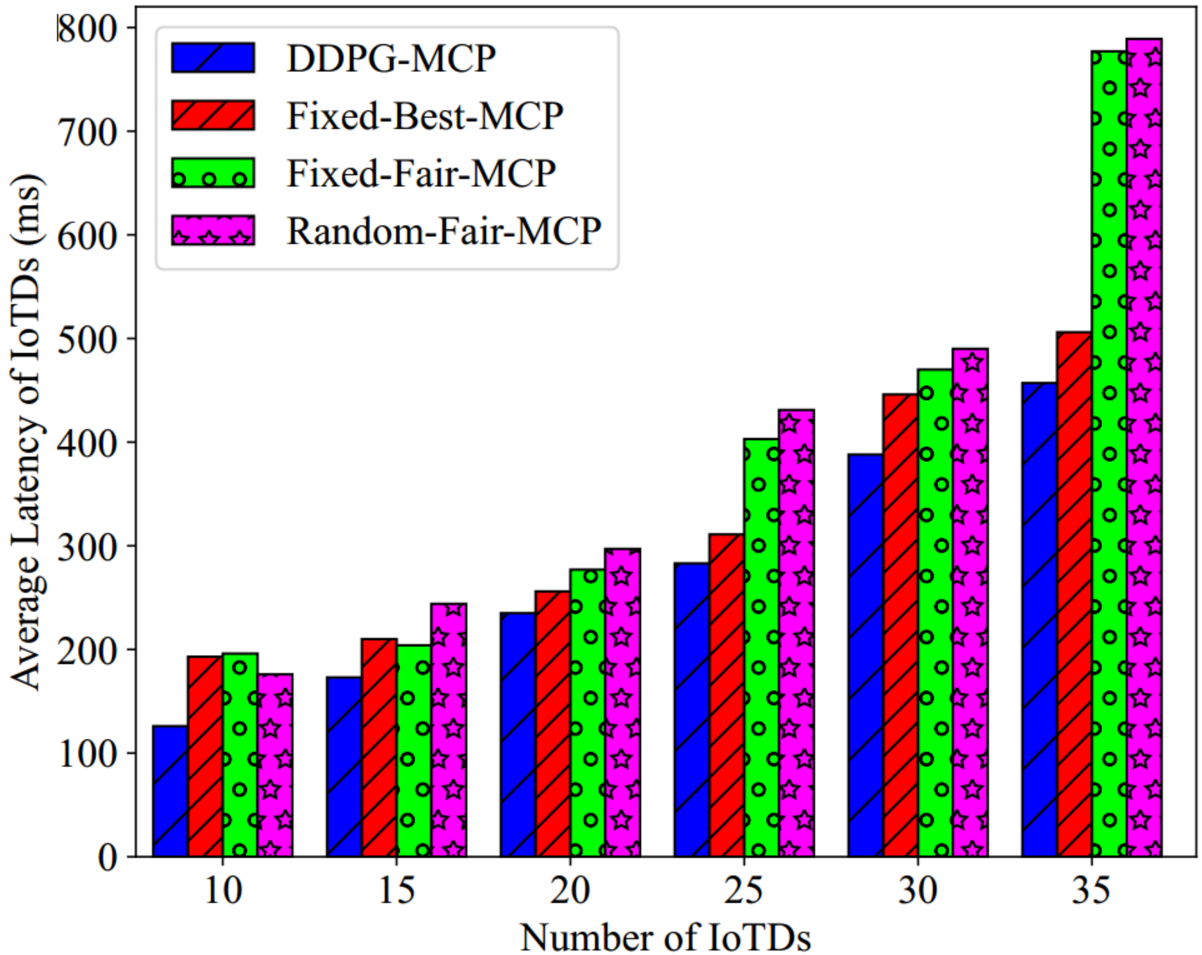
Table I: Simulation Parameters

Parameters	value
the coverage area	500 m $\times$ 500 m
$\lvert\mathcal{B}\rvert$	2
$\lvert\mathcal{U}\rvert$	$\{10, 15, \cdots, 35\}$
$\lvert\mathcal{K}\rvert$	25
r_i, the input data size	$[1, 2]$ Mb
c_i, the computing requirement	$[1, 20] \times 10^7$ CPU cycle
C_j, edge node computing capacity	2×10^{10} CPU cycle/s
$\psi_{i,j}$	$131.1 + 42.8log10(d_{i,j})$, $d_{i,j}$ in km
Rayleigh fading	8 dB
N_0	-174 dBm/Hz
P^I	20 dBm
f_j	50 RB (10 MHz)
β_0	180 kHz
t^{cloud}	100 ms

Table II: Machine Learning Set Up

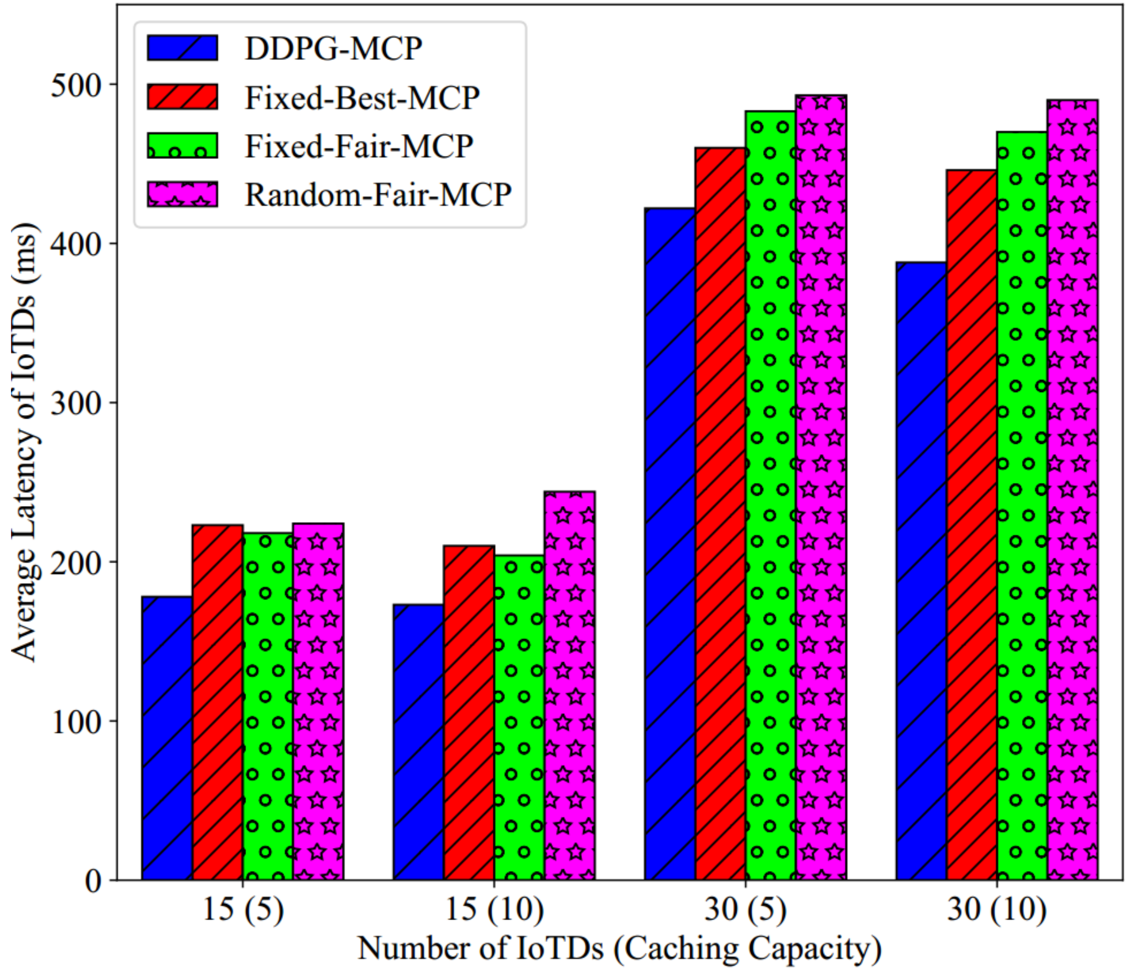
Machine Learning Parameters	
# of neurons of each hidden layer	800, 800, 800
learning rate of the actor networks	5×10^{-5}
learning rate of the actor networks	5×10^{-5}
learning rate of the critic networks	5×10^{-4}
λ, discount factor	0.99
ς, soft target update	0.001
number of training epochs	6
number of training steps	800
η^b, number of samples in a batch	64
replay buffer capacity	10^5

Evaluation Results



Average latency of served IoTDs versus # of IoTDs.

Evaluation Results (Cont'd)



Average latency of IoTDs versus caching capacity

Outline

- Caching Placement in the Edge Computing Network
- System Model and Problem Formulation
- Algorithm and Analysis
- Evaluation Results
- Conclusions

Conclusions

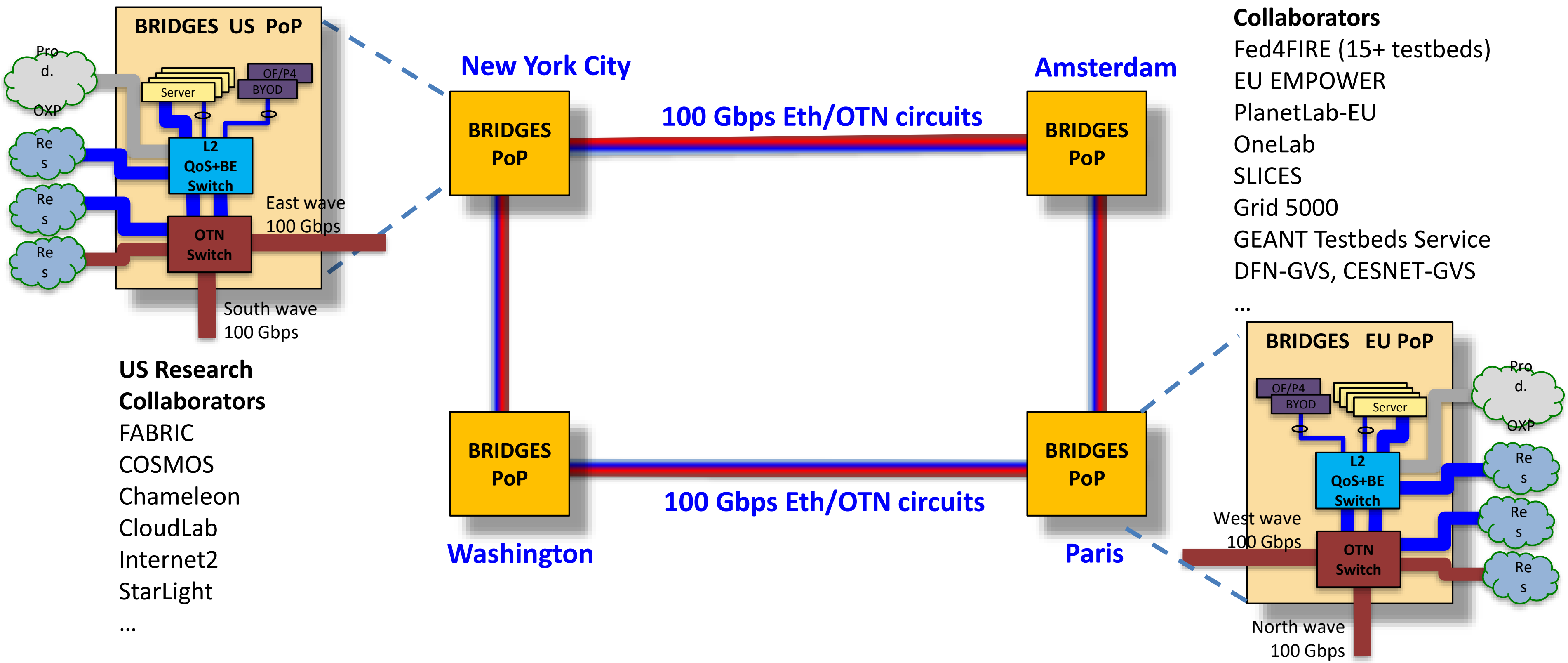
- We have studied caching in MEC networks by considering background data caching and the data collection of IoTDs. We have formulated the multi-content placement (MCP) problem in the MEC networks to minimize the average latency of IoTDs.

- A deep reinforcement learning algorithm, referred to as DDPG-MCP, is proposed to solve the MCP problem by achieving the best joint caching placement and IoTD assignment and obtaining the resource allocation through the optimal resource scheduling algorithm.

- The simulation results have demonstrated that the DDPG-MCP algorithm is superior to the baseline algorithms by up to 42% improvement for the average latency compared to baseline algorithms.

BRIDGES

A Programmable Cyber-System Funded By NSF: NSF OAC-2029221



EU Research Collaborators
Fed4FIRE (15+ testbeds)
EU EMPOWER
PlanetLab-EU
OneLab
SLICES
Grid 5000
GEANT Testbeds Service
DFN-GVS, CESNET-GVS
...

US Research Collaborators
FABRIC
COSMOS
Chameleon
CloudLab
Internet2
StarLight
...

For more information, please visit: https://cnl.gmu.edu/bridges