When Learning Joins Edge: Real-time Proportional Computation Offloading via Deep Reinforcement Learning

Ning Chen, Sheng Zhang, Zhuzhong Qian, Jie Wu, Sanglu Lu
Background

Benefits
- Low latency
- Energy efficient
- Privacy protection
- Bandwidth consumption reduction
Related work

Facets of computation offloading:

• Energy consumption  *Energy harvesting [TCCN’ 2017], Energy-Efficient [INFOCOM’ 2018]*
• Resource allocation  *SDR-AO-ST [INFOCOM’ 2017], Min-Max Fairness Guarantee [TCOM’ 2017]*
• Latency-aware scheduling  *Optimization for MECO [TWC’ 2018], uRLLC [IEEE Access’ 2018]*

Methods of computation offloading:

• NP-hard problem -> heuristic algorithm
• Minority game  *[IEEE WLC’ 2018]*
• Deep Reinforcement Learning  *[WCNC’ 2018], [IoT’ 2019]*
Motivation

Time-variant edge environment:

• Heterogeneity of mobile devices
• Fluctuation of bandwidth
• Mobility of mobile users
  (model of job arrival)
• Diversity of jobs

No “one-size-fits-all” solution:
Best algorithm depends on specific workload
Motivation

RL shows superiority in decision-making in dynamic environment

- **Agent**
  - State
  - Q network
  - Q value

- **Environment**
  - Observation
  - Reward
  - Action

Observe next state from environment
Design

Definition of action

How to transform the computation offloading problem into RL decision-making problem?
Challenge

#How to define and extract?

#Definition?

#Training methodology?
Design

Definition of state

\[ a_t = \{\text{target edge server, proportion}\} \]

Definition of state

\[ s_t = (W, r_{u1}, \ldots, r_{um}, r_{d1}, \ldots, r_{dm}, C_l, C_1, \ldots, C_m) \]

Definition of reward

The total cost of offloading workload to server \( j \) from user \( i \) is

\[ W_{ij} = \lambda \max \left( T_{ij}^{(l)}, T_{ij}^{(r)} \right) + \beta \left( E_{ij}^{(l)} + E_{ij}^{(r)} \right) \]

\( \text{reward}_1 = -W_{ij} \) or \( \text{reward}_2 = \frac{w_l - W_{ij}}{w_l} \)
Design

Training methodology (Deep Q Network)

\[ Q(S, a; \theta) \]

\[ \max_a \hat{Q}(S', a'; \theta') \]

\[ \text{Behavior network} \]

\[ \text{Target network} \]

\[ \text{Replay Buffer} \]

\[ \text{Expert Buffer} \]

\[ (S, a, r) \rightarrow (S, a, r, S') \rightarrow (S, a) \rightarrow \hat{Q}(S', a'; \theta') \rightarrow \text{Loss Function} \rightarrow \text{Training} \]

1. Supervised learning training
2. Perform gradient descent method to update $\theta$
3. Copy parameters every N steps $\theta \leftarrow \theta'$

#Update policy: $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_a Q(s', a') - Q(s, a)]$
Evaluation

Feasibility -> Superiority -> Impact factors of performance
Evaluation

Feasibility -> Superiority -> Impact factors of performance
Evaluation

Feasibility -> Superiority -> Impact factors of performance

**Baselines:**
- Random Offloading Policy
- Link Capacity Optimal Policy (LCOP).
- Computing capability Optimal Policy (CCOP).
Evaluation

Feasibility -> Superiority -> Impact factors of performance
Summary

• There are no “one-size-fits-all” solutions to solve computation offloading problem, thus we consider adopting a learning method which makes use of history experience.

• We transform the computation problem into a RL decision-making problem, and give the specific definitions of action, state and reward, respectively.

• We design a simulation to show the feasibility and superiority of our proposed ADQN, and analyze the impact factors for its performance.

• In the follow-up work, we focus to design a efficient simulator which is close to the reality.
Thank you for your listening!

Q&A