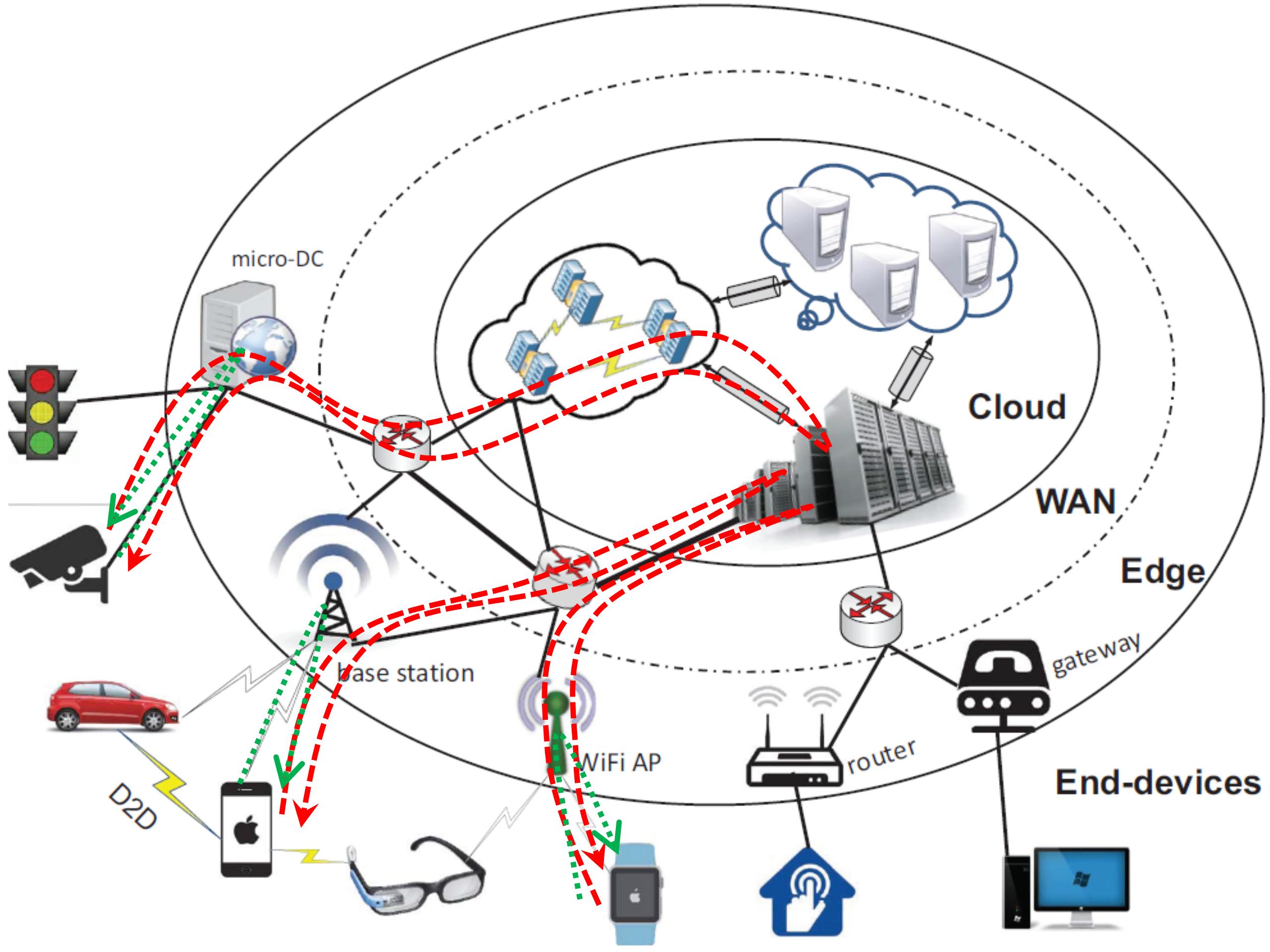
When Learning Joins Edge: Real-time Proportional Computation Offloading via Deep Reinforcement Learning

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

南京大學
NANJING UNIVERSITY

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]*

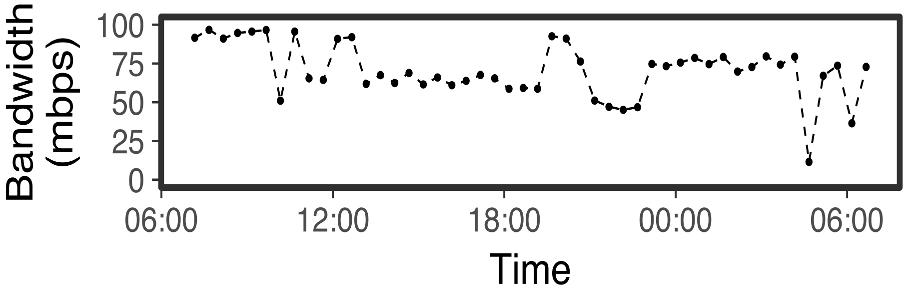
Methods of computation offloading:

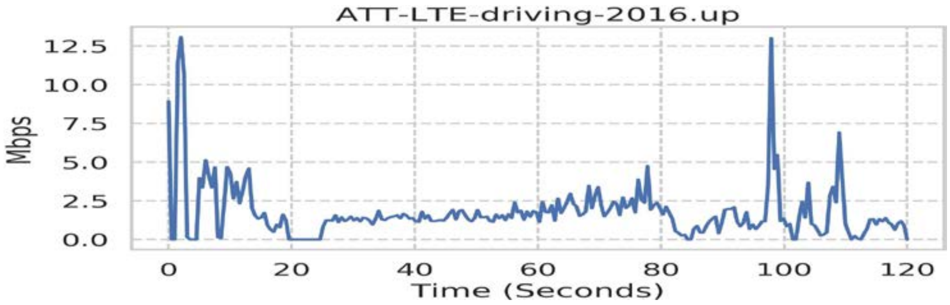
- NP-hard problem -> heuristic algorithm
- Minority game *[IEEE WLC' 2018]*
- Deep Reinforcement Learning *[WCNC' 2018], [IoT' 2019]*

Motivation

Time-variant edge environment:

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

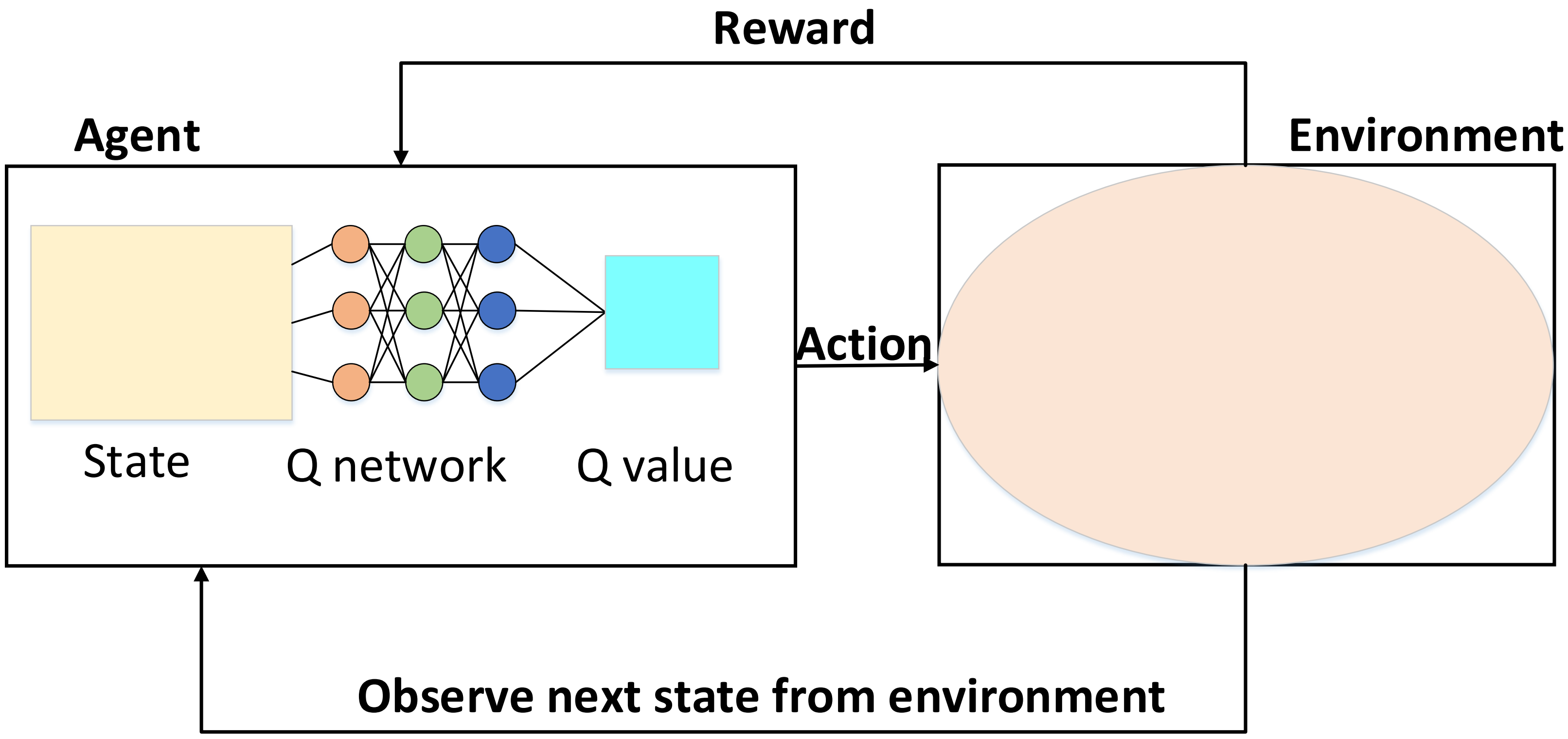




No "one-size-fits-all" solution:
Best algorithm depends on specific workload

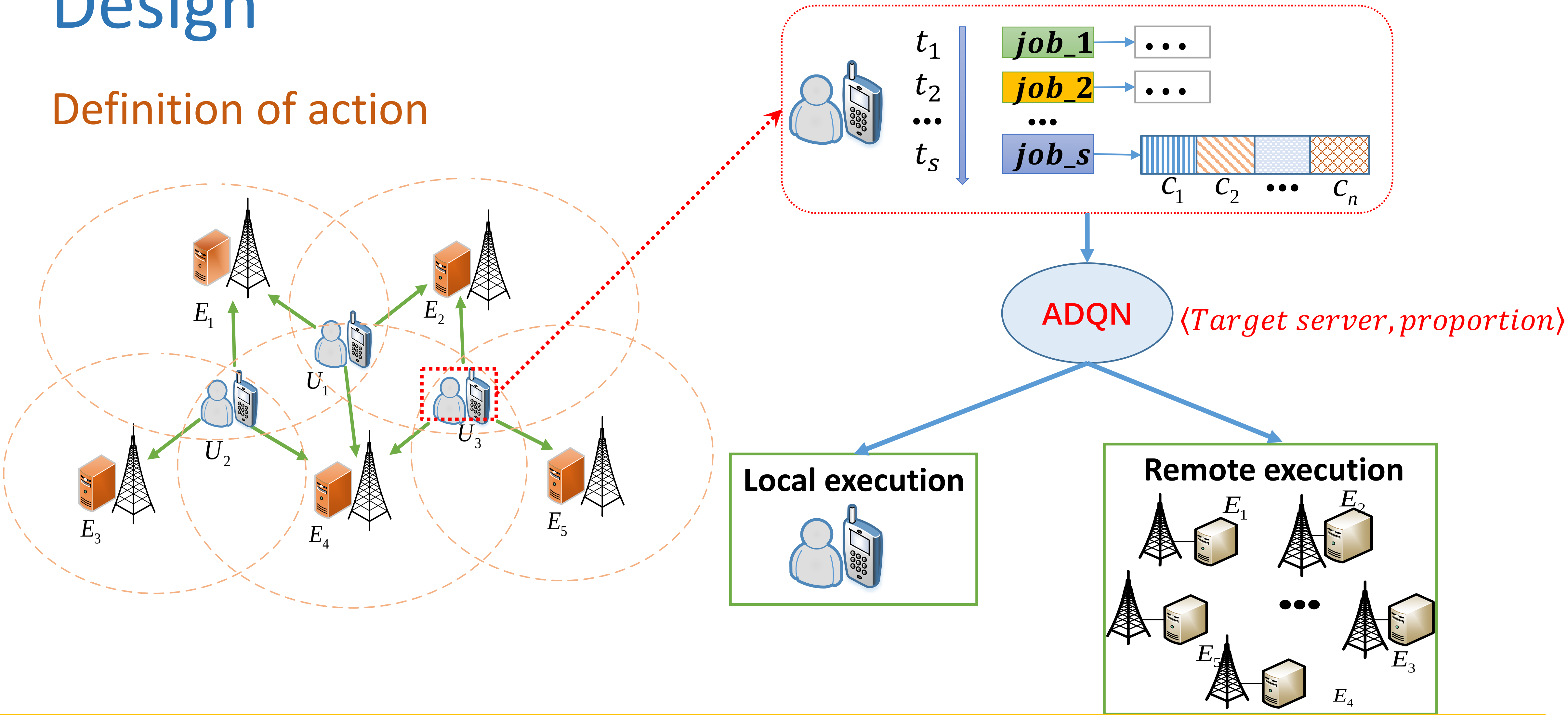
Motivation

RL shows superiority in decision-making in dynamic environment



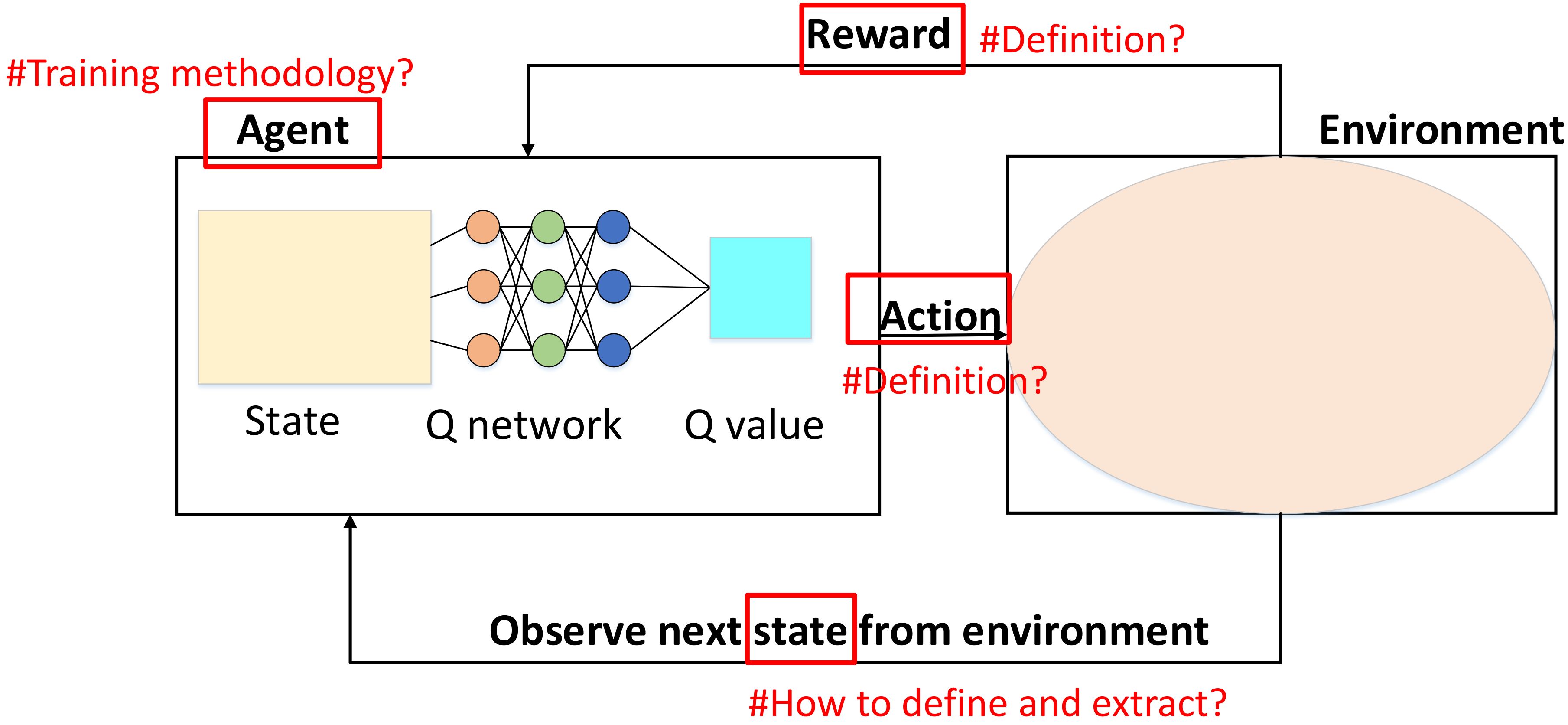
Design

Definition of action



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

Challenge



Reward
#Definition?
#Training methodology?
Agent
Environment
State
Q network
Q value
Action
#Definition?
Observe next state from environment
#How to define and extract?

Design

Definition of state

$$a_t = \langle target\ edge\ server, proportion \rangle$$

Definition of state

$$s_t = \left(\underbrace{W}_{\text{workload}}, \underbrace{r_{ul}^{1}, \cdots, r_{ul}^{m}, r_{dl}^{1}, \cdots, r_{dl}^{m}}_{\text{Network condition}}, \underbrace{C_l, C_1, \cdots, C_m}_{\text{Available resource}} \right)$$

$reward\_1 = -W_{ij}$ or

$$reward\_2 = \frac{W_l - W_{ij}}{W_l}$$

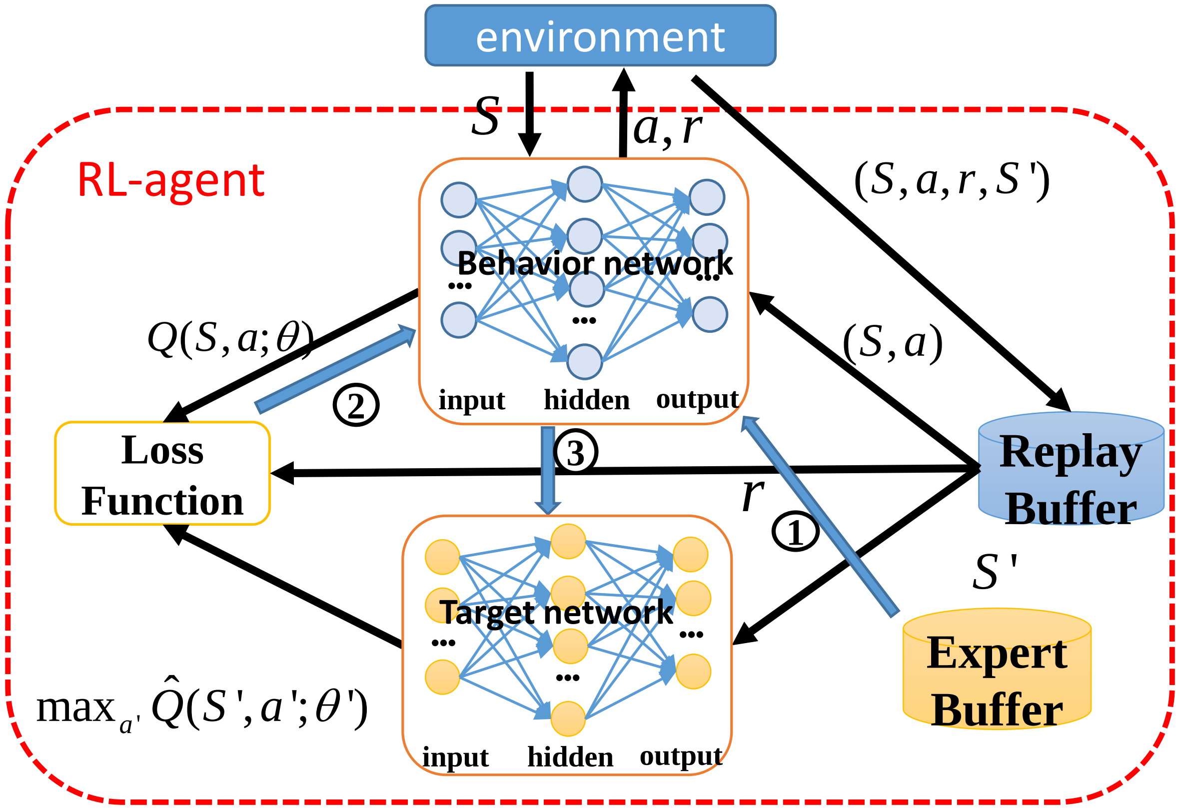
Definition of reward

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

$$W_{ij} = \underbrace{\lambda max\left(T_{ij}^{(l)}, T_{ij}^{(r)}\right)}_{\text{delay}} + \underbrace{\beta\left(E_{ij}^{(l)} + E_{ij}^{(r)}\right)}_{\text{energy}}$$

Design

Training methodology (Deep Q Network)



① Supervised learning training

② Perform gradient descent method to update θ

③ Copy parameters every N steps $\theta \leftarrow \theta'$

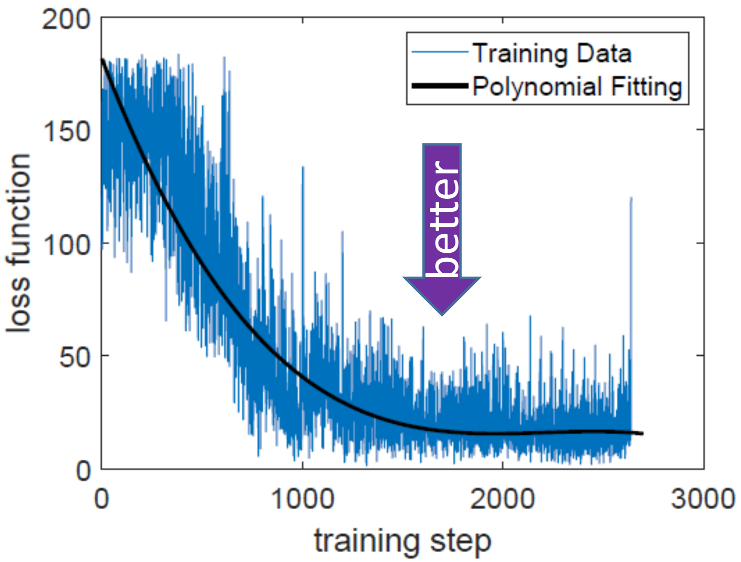
#Update policy: $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma max_{a'} Q(s',a') - Q(s,a)]$

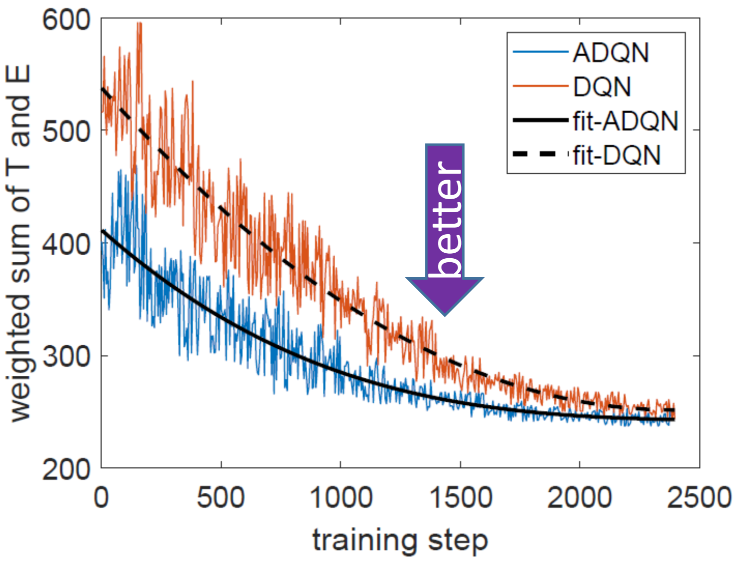
Evaluation

Feasibility -> Superiority -> Impact factors of performance

Evaluation

Feasibility -> Superiority -> Impact factors of performance



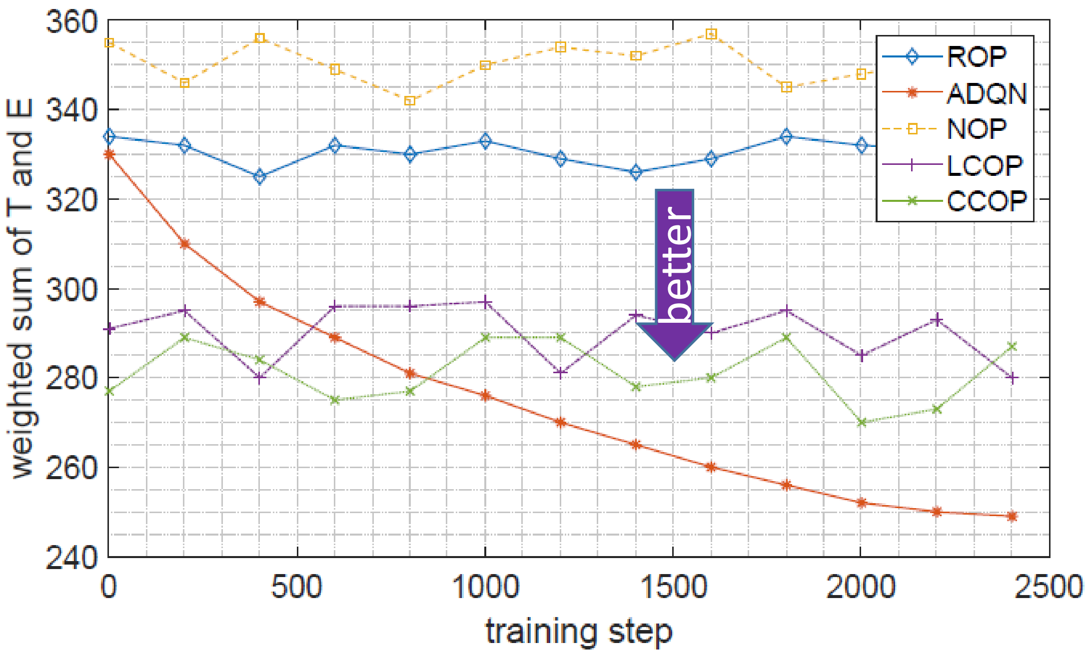


Evaluation

Feasibility -> Superiority -> Impact factors of performance

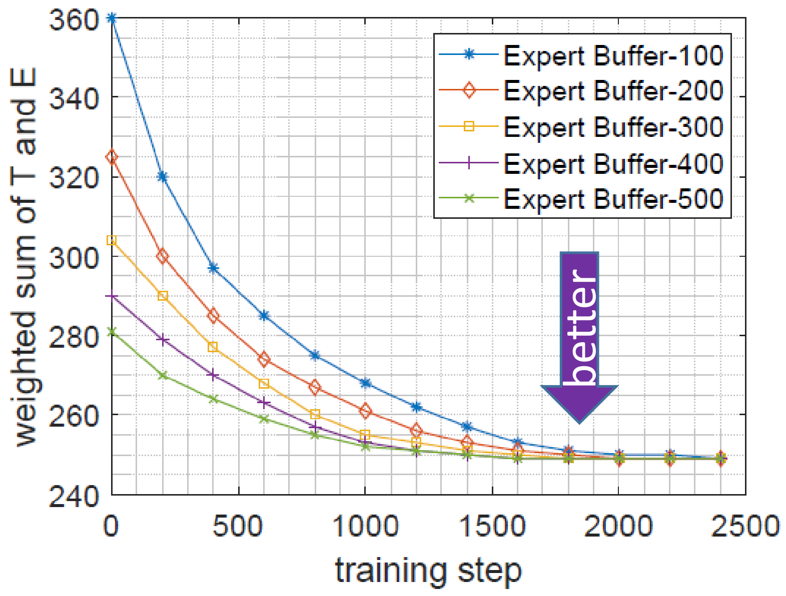
Baselines:

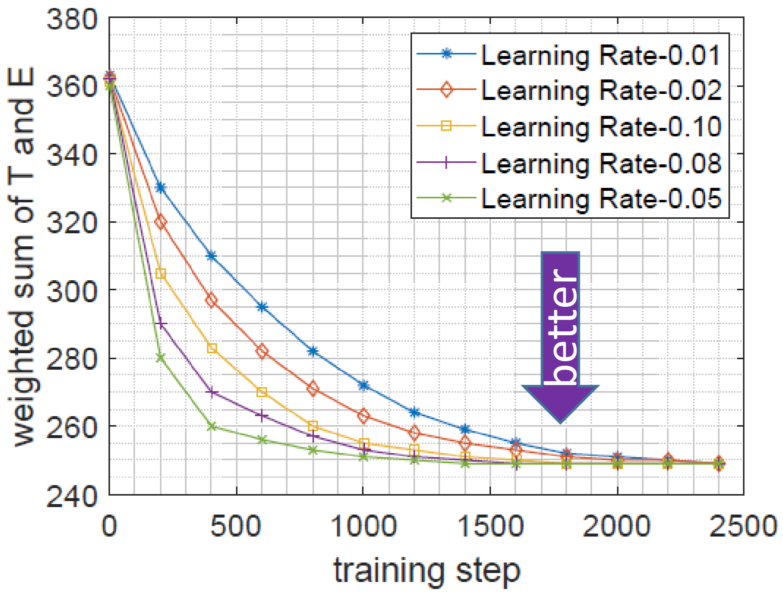
- Random Offloading Policy
- Link Capacity Optimal Policy (LCOP).
- Computing capability Optimal Policy (CCOP).



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