# CS565: Cloud Computing

## Reading Assignment 1

**Name**: M. Maheeth Reddy **Roll No.**: 1801CS31 **Date**: 18-Apr-2022

---

**Task**: **Summarize Section 4 of the given research paper**

## Energy-efficient UAV-enabled computation offloading for industrial internet of things: a deep reinforcement learning approach

**Shuo Shi • Meng Wang • Shushi Gu • Zhong Zheng**

### Summary of Section 4 - Simulation results

**4.1 Details of Parameters settings**

Consider the following smart factory scenario:

1. Area is 800m x 800m (fixed)
2. t is the time interval at which devices need to deal with tasks generated by control center
3. Tasks can be either offloaded to UAVs or to the remote cloud center or execute locally.
4. No. of IIoT devices M is 1000
5. No. of UAVs is 5
6. Flying height H is 100m
7. Communication Range of each UAV R is 300m
8. Devices are randomly distributed in the fixed area with local computation capablities $C_l$ = 500 MHz.
9. The computation capabilities of UAVs $C_u$ is 2 Ghz
10. The cloud center $C_c$ is 100 GHz
11. The sizes of tasks are distributed within [100 Kb,100 Mb], the CPU circles required by tasks are generated randomly within [$5 \times 10^5$, $5 \times 10^9$]
12. The tolerant delay of tasks are chosen in [0.01s, 1s]
13. LoS channel power gain $\beta_0$ is $1.42 \times 10^{-4}$ for the reference distance 1m
14. Bandwidth B is 15 MHz, w is 10MHz
15. Transmission power of MU to UAVs $P_i$ is 0.01 W
16. Transmission power of MU to base stations Pi is 0.015 W and the noise power $\sigma^2$ is -90dBm/Hz
17. The propagation time factor from MU to cloud server $\tau$ is $4.0 \times 10^{-9}$ s/bit
18. The local energy consumption factor $\theta$ is set as $1 \times 10^{-23}$ J/cycle.
19. A two-layer fully-connected neural network is also used:
20. with each layer 100 neuron units as out training network
21. Learning rate $\alpha$ = 0.0001
22. Memory size of experiences = 10000
23. Batch size = 50
24. Target replace steps Z = 500
25. $\epsilon$ is set as 0.9
26. the reward decay $\Upsilon$ = 0.7

Note: All simulations are run in Python 3.7 with Tensorflow 1.3 on Windows 10.

### 4.2 Performance analysis

Convergence performance of the MDSPR:

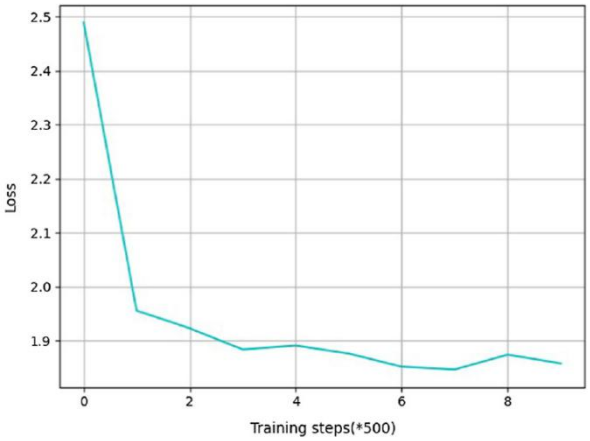


Fig 4

Notice the change in loss from Fig 4. The loss will reduce rapidly at the beginning because the agents take actions randomly and the approximation is not precise. But after 100 steps the loss become stable, which indicates the convergence of MDSPR.

Comparison of performance of MDSPR with benchmark algorithms (random algorithm, DQN algorithm and DDQN algorithm):

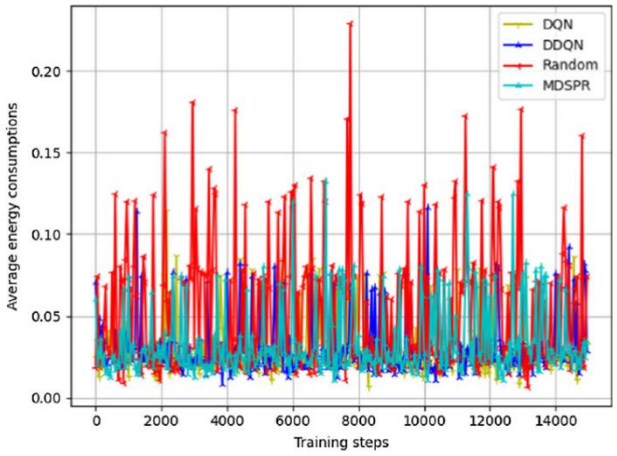


Fig 5

Notice the average energy consumption of a mobile user when dealing with a single task from Fig 5. This is because the tasks are generated with random $s_i$; $c_i$; $d_i$, and an inappropriate offloading decision may cause huge additional energy consumption.

Consider a task with high computation requirement and small size. It will require significantly higher cost if executed locally when compared to being offloaded to the remote cloud center.

In the first n training steps, the agent makes offloading decisions according to greedy policy, after the learning stage, the average energy consumption become stable due to the convergence of neural networks.

Notice the successful task execution ratio versus training steps from Fig 6. We consider that the execution of a task has failed in the following two cases:

1. **The execution time exceeds the tolerant time $d_i$**: This means the task cannot be finished in the required QoS.
2. **UAVs' offloading**: If a task is chosen to offload to a UAV, and the distance between the mobile user and the target UAV is larger than the communication range of UAVs.

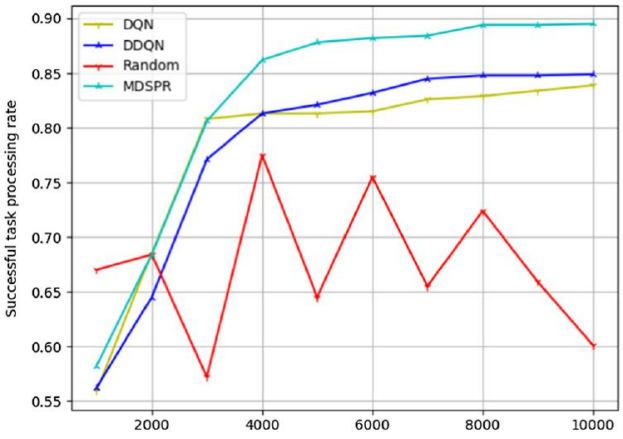


Fig 6

The successful task execution ratio of MDSPR outperforms the benchmark algorithms, which indicates MDSPR is more probable to make energy-efficient decisions.

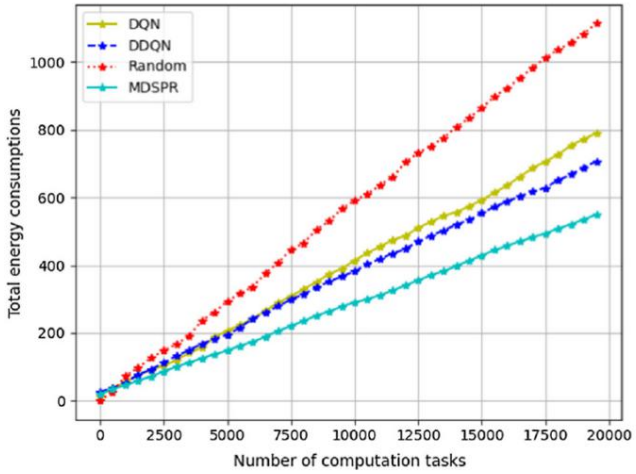


**Fig 7**

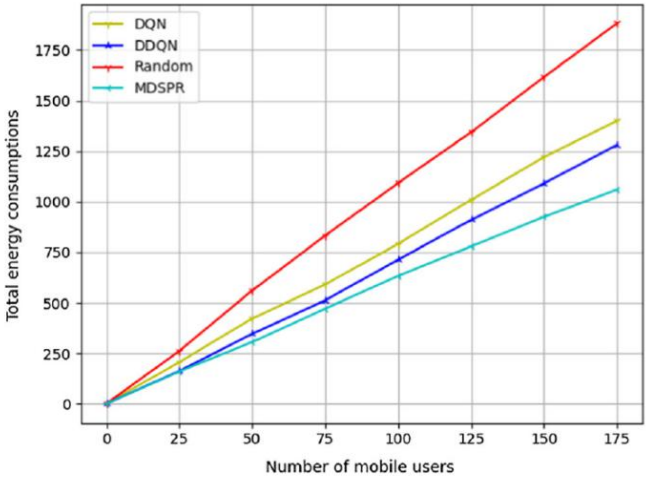


**Fig 8**

From Fig. 7, it is evident that MDSPR has better total energy consumption than the three benchmark algorithms, because MDSPR chooses more valuable experiences to learn in each training step, thus maximizing the overall long-time reward.

Also its energy consumption increases almost linearly with the increasing tasks because the max Q-value is used by the agents to make offloading decisions upon each task's arrival.

After the convergence of the neural network, the action with highest Q-value has a high probability to obtain minimize energy consumption.

Notice that MDSPR also outperforms the benchmark algorithms in terms of the total energy consumption upon different no. of devices (Given the states of devices are initialized randomly at the beginning and each device processes 500 different tasks). (See Fig 8)

The time complexity of MDSPR algorithm w.r.t mobile users is n. This implies that the the total energy consumption increase linearly with no. of mobile users. So, MDSPR algorithm enables different users to make decisions in a distributed manner, with the possibility of being applied in large-scale scenarios.