北京邮电大学 本科毕业设计（论文）初期进度报告

Project Early-term Progress Report

学院 School	International School	**专业 Programme**	**e-Commerce Engineering with Law**		
姓 Family name	Ke	**名 First Name**	Jiaming		
BUPT 学号 BUPT number	2019213499	**QM 学号 QM number**	190894179	**BUPT 学号 BUPT number**	2019213499
论文题目 Project Title	Meta Reinforcement Learning for Task Offloading in MEC Networks				

已完成工作 Finished work:

Introduction

Nowadays, mobile devices are responsible for many compute intensive tasks, such as video or image processing and data analysis. For example, a smartphone app that performs image recognition or data analysis can be easily downloaded and used by millions of people around the world. Despite the continuous development of mobile devices, due to limited computing resources, these devices may not be able to process all tasks locally with low latency. In order to promote efficient task processing, mobile edge computing(MEC) is introduced.

In practical dynamic scenarios, the MEC environment is often affected by many factors anytime and anywhere. Conventional intelligent algorithms are usually based on neural networks. When the MEC environment changes, its original parameters will all fail and a large amount of training data is required to train from scratch, which makes the learning efficiency low. Such repeated training will consume resources and weaken the performance of the MEC system. At the same time, in order to improve efficiency, high configuration equipment is also required to adapt to high-intensity training. Considering the delay and energy consumption of IoT, offloading decisions can be made for a workflow with a series of dependent tasks.

However, this kind of problem is generally NP-hard, traditional optimization methods are difficult to achieve results efficiently.

Literature review

One promising way of addressing the above issue is to bring deep learning techniques (especially DRL methods) into the computing paradigm of edge-cloud collaboration.[1] Tang an Wong incorporate the long short-term memory (LSTM), dueling deep Q-network (DQN), and double-DQN techniques. Simulation results with 50 mobile devices and five edge nodes show that the proposed algorithm can reduce the ratio of dropped tasks and average task delay. Unfortunately, conventional DRL algorithms have the disadvantage of slower learning speed, which is mainly due to the weak inductive bias. A learning procedure with weak inductive bias will be able to adapt to a wide range of situations, however, it is generally less efficient.

In the context of edge-cloud computing, meta reinforcement learning (meta-RL) can be used to optimize the decision of where to offload computation tasks. In this context, meta-RL refers to the use

of reinforcement learning techniques to learn a policy that can adapt to different situations and make decisions about task offloading that will maximize some reward signal.
One possible application of meta-RL for task offloading in edge-cloud computing is to optimize the trade-off between the cost of offloading computation to the cloud and the benefits of doing so in terms of reduced latency and increased performance. For example, a mobile device with limited computational resources might use meta-RL to learn a policy for deciding whether to perform a computation locally or to offload it to the cloud, based on factors such as the current battery level, network conditions, and the expected cost and benefits of offloading. Another possible application of meta-RL for task offloading in edge-cloud computing is to optimize the allocation of tasks across a network of edge devices and cloud resources.

Qu's team proposed a Deep Meta Reinforcement Learning-based Offloading (DMRO) algorithm,[2] which combines multiple parallel DNNs with Q-learning to make fine-grained offloading decisions. Through several simulation experiments, DMRO compared with traditional Deep Reinforcement Learning (DRL) algorithms, the offloading effect of DMRO can be improved by 17.6%. In addition, the model has strong portability when making real-time offloading decisions, and can fast adapt to a new MEC task environment. Another paper proposed a new MRLCO,[3] focused on fast adaptation to dynamic offloading scenarios. Includes a new idea to model the dynamic computation offloading process as multiple MDPs, where the learning of offloading policies is decomposed into two parts: effectively learning a meta policy among different MDPs, and fast learning a specific policy for each MDP based on the meta policy. Convert the offloading decision process as a sequence prediction process and design a custom seq2seq neural network to represent the offloading policy. An embedding method is also proposed to embed the vertices of a DAG. The results show that MRLCO achieves the lowest latency within a small number of training steps compared to three baseline algorithms.

Some recent research in this area has focused on developing meta-RL algorithms that can adapt to changing environments or tasks, such as those that involve dynamic resource constraints or uncertain demand. Other research has focused on improving the sample efficiency of meta-RL algorithms,[4] so that they can learn more quickly and effectively from limited data.

Overall, the use of meta-RL for task offloading has the potential to improve the efficiency and performance of distributed systems by allowing them to adapt and optimize their behavior in real-time. However, further research is needed to address challenges such as scalability and robustness to ensure that meta-RL algorithms can be applied effectively in real-world systems.

References:

[1] Ming Tang, Vincent W.S. Wong. Deep Reinforcement Learning for Task Offloading in Mobile Edge Computing Systems. arXiv:2005.02459 [cs.NI]

[2] G. Qu, H. Wu, R. Li and P. Jiao, "DMRO: A Deep Meta Reinforcement Learning-Based Task Offloading Framework for Edge-Cloud Computing," in *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3448-3459, Sept. 2021, doi: 10.1109/TNSM.2021.3087258.

[3] J. Wang, J. Hu, G. Min, A. Y. Zomaya and N. Georgalas, "Fast Adaptive Task Offloading in Edge Computing Based on Meta Reinforcement Learning," in IEEE Transactions on Parallel and Distributed Systems, vol. 32, no. 1, pp. 242-253, 1 Jan. 2021, doi: 10.1109/TPDS.2020.3014896.

[4] T. Alfakih, M. M. Hassan, A. Gumaei, C. Savaglio, and G. Fortino, “Task Offloading and Resource Allocation for Mobile Edge Computing by Deep Reinforcement Learning Based on SARSA,” IEEE Access, vol. 8, pp. 54074–54084, 2020, doi: 10.1109/access.2020.2981434.

[5] A. Feriani and E. Hossain, "Single and Multi-Agent Deep Reinforcement Learning for AI-Enabled Wireless Networks: A Tutorial," in IEEE Communications Surveys & Tutorials, vol. 23, no. 2, pp. 1226-1252, Secondquarter 2021, doi: 10.1109/COMST.2021.3063822.

是否符合进度？On schedule as per GANTT chart?
YES

下一步 Next steps:

How to formulate a MEC network is a problem. We have to work out 3 issues: setting up the communication model and task scheduling system, and formulating a Markovian optimization problem, finally implementing the code for Meta Learning algorithms on the edge nodes.

Preliminary model

The mobile edge computing environment for using Meta Learning on each node for distributed Task Offloading would consist of the following components:

1. Mobile devices: These are the devices that are connected to the edge network and would be used to send and receive data and tasks.
2. Edge nodes: These are the servers or devices located at the edge of the network that are responsible for processing tasks and data. They would be equipped with Meta Learning algorithms to enable them to learn and adapt to the tasks they are processing.
3. Communication infrastructure: This would consist of the network infrastructure (e.g. routers, switches, etc.) that enables the mobile devices and edge nodes to communicate with each other.
4. Task scheduling and distribution system: This system would be responsible for distributing tasks to the appropriate edge nodes based on their capabilities and workload. It would also monitor the progress of the tasks and ensure that they are completed efficiently.

Then to transform the problem of task offloading into a Markovian process, we can first define the set of possible states that the system can be in. These states should capture all the relevant information about the system that can affect the decision of which task to offload.
For example, if we are considering a mobile device that can offload tasks to the cloud or to a nearby edge server, the states could include:

- The current battery level of the mobile device
- The current network conditions (e.g. bandwidth, latency, cost)
- The current workload of the mobile device
- The current workload of the cloud and edge servers
- The current cost of offloading tasks to the cloud or edge servers

Once we have defined the set of states, we can define the transition probabilities between these states. The transition probabilities represent the likelihood of transitioning from one state to another based on the actions taken by the system.