Learning and Forgetting for Server Selection in Mobile Edge-Computing: A Perspective

Andrea Ortiz

Institute of Telecommunications, TU Wien.

Corresponding author(s). E-mail(s): andrea.ortiz@tuwien.ac.at;

Abstract

Mobile Edge Computing (MEC) is an architecture that brings computational capabilities to the edge of mobile networks. It enables low-latency and efficient task execution by allowing the Mobile Units (MU) to offload computation tasks to nearby computing servers. However, server selection in MEC remains an open and complex challenge due to the non-stationary nature of the system dynamics, where channel conditions, server loads, and resource availability change over time. Traditional reinforcement learning approaches, while effective for adaptive decision-making, often assume stationary system dynamics. In this perspective, we discuss the role of learning and forgetting mechanisms in MEC server selection, emphasizing the need for adaptive mechanisms that can retain and exploit relevant experience while discarding outdated information.

Keywords: Mobile Edge Computing, Server Selection, Reinforcement Learning, Non-stationary systems

1 Introduction

In recent years, Mobile Edge Computing (MEC) has established itself as an effective architecture to bring computing and storage capabilities to the edge of the network. In traditional mobile networks, the Mobile Units (MUs) use wireless communication links to the radio access network. The radio access network is formed by Base Stations (BSs) and Access Pointss (APs) using different technologies and covering areas of different sizes, as shown in Fig. 1. The BSs and APs are connected to the core network and the Internet via wired or high-capacity wireless links, e.g., using fiber optics or millimeter wave technology. The connection to the core network also enables the provision of cloud computing services that allow the MUs to offload the execution

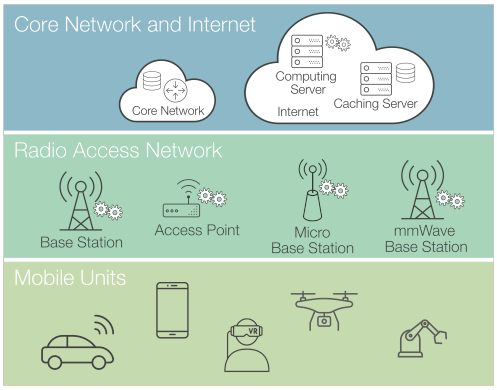

Fig. 1 Main components of a conventional mobile network.

of computationally expensive tasks. Examples of such tasks are health data monitoring [**?**], gesture and face recognition [**?**, **?**], virtual and augmented reality [**?**, **?**], as well as the synchronization of digital twin models [**?**, **?**]. However, for low-latency and high-reliability services, such as health monitoring [**?**], vehicular communications [**?**] or Internet of Things (IoT) [**?**], offloading tasks to the cloud could violate their strict latency and reliability requirements. This is due to the fact that cloud computing relies on data centers that are located far away from the MUs, thus introducing significant delays, congestion, and unpredictable latency variations [**?**, **?**]. To overcome these problems, in MEC, the BSs and APs are equipped with computing and caching capabilities. As a result, computing services are closer to the MUs and the latency in the execution of tasks is reduced. Furthermore, MEC improves the system's reliability by distributing processing loads across the different BSs and APs, minimizing the dependence on a single centralized cloud, and enabling real-time decision making with improved fault tolerance. This translates in faster computations, better quality of experience and more flexibility for network operators and application providers.

The performance of MEC highly depends on the association between MUs, BSs and APs, and the task offloading schedule, i.e., on the selection of the BS or AP to be used by each MU for offloading, and the selection of which task to offload at each time. These decisions are non-trivial due to the dynamic nature of MEC. The quality of the wireless channels between BSs and MUs vary due to the mobility of the MUs. Furthermore, the availability of computing resources at the BS depends on the number of offloading MUs and the particular demands of each task. Moreover, the server selection and offloading decisions need to be resilient and enable fast adaptation to abrupt and unforeseen changes like failures and anomalies, temporary blockages or rapid demand surges.

In this paper, we discuss the problem of BS and AP selection in MEC systems. Specifically, we consider a MU who sequentially selects BSs and APs for task offloading. The MU decides whether to use the computing capabilities at the edge or to take advantage of the computing services of the cloud. We argue that in order to enable fast adaption to abrupt changes in the system, and minimize the latency and energy

consumption of the MUs, the BS and AP selection policy at the MUs must strike a balance between *learning* and *forgetting* mechanisms. Learning allows the MU to exploit past decisions to find the optimal offloading policy according to the network conditions. Forgetting helps the MU to adapt to non-stationary changes, like anomalies, by selectively discarding outdated and irrelevant data. The challenge in balancing these two mechanisms lies in the fact that in both of them exists the risk of not fulfilling the strict requirements of the task, thus compromising the system performance.

2 State of the Art

Research effort has been put into finding BS selection and offloading strategies for MEC. Numerical optimization and game theoretical methods are used in [**?**, **?**, **?**] under the assumption of complete a-priori knowledge of the system dynamics. These works aim at minimizing the latency and the energy consumption of MEC systems. As the resulting problems are often non-convex or even NP-hard, these works investigate the use of relaxation techniques and heuristic approaches for their solution. The main advantage of using optimization as well as game theoretical methods is that they enable the provision of theoretical performance bounds. However, they are usually computationally expensive and require information which is not known in advance, which makes them unsuitable for real-time MEC applications.

In order to investigate dynamic MEC systems, recent works have considered the use of reinforcement learning techniques [**?**, **?**, **?**, **?**, **?**]. Although these works are able to overcome the unrealistic assumption of complete a-priori knowledge of the system dynamics, they still rely on simplified assumptions about the characteristics of the dynamic behaviour of MEC systems. For instance, [**?**, **?**] assume the MEC system is dynamic but statistically stationary, [**?**] assumes identical computational capabilities across BSs, in [**?**] the load variations in the BSs are modeled as a non-stationary random process while the quality fluctuations of the wireless channels are ignored and in [**?**] the dependency of the BS computing capabilities on the tasks characteristics is not considered. In [**?**], a first step into including forgetting mechanisms in MEC is presented. The authors model the dynamics of the MEC system as statistical non-stationary and use a fixed time window to detect changes in the average performance and increase the adaptation speed. Nevertheless, [**?**] focuses on the interaction of BSs and MUs and disregards the contribution of cloud computing services.

3 Learning and Forgetting in MEC

3.1 Why Learning?

In MEC, the MU repeatedly decides whether to perform its tasks locally, to offload them to a BS or AP, or to offload them to a cloud server. The goal of the MU is to minimize a cost function, whose definition depends on the particular requirements of each MEC application. Nevertheless, in most cases the cost function is defined as a linear combination of the latency experienced by the MU and the energy it consumes. In case the task is performed locally, the latency includes the processing and queuing latency at the MU. Similarly, the energy corresponds to the processing required for the

task execution and the one consumed in idle mode. In case the task is offloaded, the experienced latency includes the transmission latency for the communication between the MU and the selected computing server, i.e., BS, AP or cloud server, as well as the processing, queuing and migration latencies at the MU and the computing server. The energy consumed by the MU includes in this case, the transmission energy and the energy consumed in idle mode.

As discussed in Sec. 2, MEC systems are dynamic due to fluctuations in the wireless channel conditions, the mobility of the MUs, and the varying loads of the BSs, APs and cloud servers. Therefore, policies for the selection of the computing servers that can adapt to these variations without compromising the system's performance are needed. Reinforcement learning is a suitable tool to obtain such policies[1]. Using reinforcement learning, the MU learns, in an online or an offline fashion, the optimal server selection policy through sequential decision-making. Specifically, the MU builds estimates of the expected cost associated to offloading tasks to each possible computing server and the expected cost of local computation. The estimates are built by repeatedly selecting the different computation options, i.e., offloading to a computing server and local computation, for task execution and evaluating their performance, e.g., the cost. In reinforcement learning terminology, MU balances exploration, i.e., trying different computing servers as well as local computation to evaluate their performance, and exploitation, i.e., choosing the option that has provided the lowest average cost based on past experiences. This trade-off ensures that the MU can both discover potentially better options and leverage the best-known strategy for optimal performance.

3.2 Why Forgetting?

Although using reinforcement learning for computing server selection in MEC enables the adaptation to dynamic scenarios, traditional reinforcement learning approaches rely on the assumption that the systems changes in a statistically stationary manner. This means that the expected cost associated to selecting each of the BSs, APs, cloud servers or even local computation remains constant over time. Under such assumption, the MU increases the rate of exploitation as time progresses to ensure the optimal computing server is chosen often. However, in realistic scenarios, anomalies and failures can violate the stationarity assumption and change the expected cost associated to each computing server. In this setting, traditional offline and online reinforcement learning approaches struggle to adapt to the changes in the expected cost. On the one hand, when offline reinforcement learning is used, the MU's offloading policy is fixed and cannot be changed through real-time updates. On the other hand, when online reinforcement learning approaches are considered, the adaptation is usually slow because the rate of exploration is lower, compared to the initial learning phase, and the MU tends to stick with the exploitation of what it has deemed as the best option.

To enable fast adaptation in statistically non-stationary environments, the MU must be equipped with forgetting mechanisms that allow it to discard outdated information, e.g., outdated estimates of the cost, and prioritize recent experiences when making offloading decisions. In this way, the MU is able to continuously adjust its

[1]Note that although we focus on computing server selection for task offloading, reinforcement learning can also be used for resource allocation in MEC.

server selection and task offloading strategy in response to changes in the channel conditions, server loads, and task requirements, thus preventing overfitting to past system conditions. The main challenge when introducing forgetting mechanisms in MEC is the definition of outdated. A strict forgetting mechanism might lead to premature loss of valuable experiences, hindering long-term learning, while a relaxed forgetting mechanism could rely too much on outdated information and reduce the MU's adaptation capabilities. Furthermore, as the non-stationary behavior of MEC systems is caused by multiple factor, the rate at which their probability distributions vary might differ from each other. As a results, context-aware forgetting mechanisms are required.

Common forgetting mechanisms for reinforcement learning include sliding window experience replay, decay-based forgetting, adaptive decay-forgetting, priority-based forgetting and change-point detection. Sliding window experience replay uses a fixed-size temporal window to decide which experiences to use for learning and which ones to discard. Decay-based and adaptive decay forgetting use a discount factor to gradually reduce the contribution of outdated experiences in the learning. In the former case, the discount factor is fixed. For the latter, it is dynamically adjusted depending on the context, e.g., the system conditions. Instead of discarding outdated experiences uniformly or randomly, priority-based forgetting assigns an importance score to each stored experience based on factors like the magnitude of the achieved reward, the temporal relevance or the overall contribution to learning. Experiences with low priority, for example, outdated or less useful for making offloading decisions are removed. Change-point detection mechanisms aim at actively looking for changes in the distribution of the expected cost in order to trigger exploration at MU. In this way, past experiences are not discarded until a significant shift in the system's dynamics is detected, ensuring that valuable experiences are retained when conditions remain stable. The change-point detection can be done by monitoring variations in metrics such as latency, server load, or transmission rates to help the MU differentiate between natural fluctuations and true distributional changes. Once a change is detected, the MU can adjust its exploration-exploitation balance, prioritizing the collection of new experiences to learn an updated server selection strategy. Although steps have been taken to introduce forgetting mechanisms in MEC, see for example [**?**], research effort is still needed to identify the main contributing factors for non-stationary behavior in MEC and to develop adaptive forgetting mechanism that balance performance maximization with fast reaction to changes. Future research should also focus on scalability, computational efficiency, and real-time implementation of forgetting mechanisms to ensure practical deployment in dynamic MEC systems.

Acknowledgements. This work has been funded by the Vienna Science and Technology Fund (WWTF) [Grant ID: 10.47379/VRG23002].

References

[1] Qiang He, Zhaolin Xi, Zheng Feng, Yueyang Teng, Lianbo Ma, Yuliang Cai, and Keping Yu. Telemedicine monitoring system based on fog/edge computing: A survey. *IEEE Transactions on Services Computing*, 18(1):479–498, 2025.

[2] Jiagang Liu, Ju Ren, Yongmin Zhang, Xuhong Peng, Yaoxue Zhang, and Yuanyuan Yang. Efficient dependent task offloading for multiple applications in MEC-cloud system. *IEEE Transactions on Mobile Computing*, 22(4):2147–2162, 2023.

[3] Hongbo Jiang, Xingxia Dai, Zhu Xiao, and Arun Iyengar. Joint task offloading and resource allocation for energy-constrained mobile edge computing. *IEEE Transactions on Mobile Computing*, 22(7):4000–4015, 2023.

[4] Latif U. Khan, Walid Saad, Dusit Niyato, Zhu Han, and Choong Seon Hong. Digital-twin-enabled 6g: Vision, architectural trends, and future directions. *IEEE Communications Magazine*, 60(1):74–80, 2022.

[5] Maximilian Wirth, Andrea Ortiz, and Anja Klein. Risk-aware bandits for digital twin placement in non-stationary mobile edge computing. In *IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 13–18, 2024.

[6] Arash Bozorgchenani, Setareh Maghsudi, Daniele Tarchi, and Ekram Hossain. Computation offloading in heterogeneous vehicular edge networks: On-line and off-policy bandit solutions. *IEEE Transactions on Mobile Computing*, 21(12):4233–4248, 2021.

[7] Thinh Quang Dinh, Jianhua Tang, Quang Duy La, and Tony Q. S. Quek. Offloading in mobile edge computing: Task allocation and computational frequency scaling. *IEEE Transactions on Communications*, 65(8):3571–3584, 2017.

[8] Jianhui Liu and Qi Zhang. Offloading schemes in mobile edge computing for ultra-reliable low latency communications. *IEEE Access*, 6:12825–12837, 2018.

[9] Hong Kang, Minghao Li, Lehao Lin, Sizheng Fan, and Wei Cai. Bridging incentives and dependencies: An iterative combinatorial auction approach to dependency-aware offloading in mobile edge computing. *IEEE Transactions on Mobile Computing*, 2024.

[10] Jiawen Chen, Yajun Yang, Chenyang Wang, Heng Zhang, Chao Qiu, and Xiaofei Wang. Multitask offloading strategy optimization based on directed acyclic graphs for edge computing. *IEEE Internet of Things Journal*, 9(12):9367–9378, 2021.

[11] Tao Ouyang, Rui Li, Xu Chen, Zhi Zhou, and Xin Tang. Adaptive user-managed service placement for mobile edge computing: An online learning approach. In *IEEE International Conference on Computer Communications (Infocom)*, pages 1468–1476. IEEE, 2019.

[12] Saeed Ghoorchian and Setareh Maghsudi. Multi-armed bandit for energy-efficient and delay-sensitive edge computing in dynamic networks with uncertainty. *IEEE Transactions on Cognitive Communications and Networking*, 7(1):279–293, 2020.