DEEP REINFORCEMENT LEARNING BASED ADMISSION CONTROL FOR THROUGHPUT MAXIMIZATION IN MOBILE EDGE COMPUTING

by

Yitong Zhou

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
June 2021

This thesis entitled ***"Deep Reinforcement Learning Based Admission Control for Throughput Maximization in Mobile Edge Computing"*** *by* ***Yitong Zhou*** *is for the partial fulfillment of the requirements for the degree of* ***Masters of Computer Science****.*

Table of Contents

List of Tables

List of Figures

Abstract

With the development of wireless network technologies, such as LTE/5G, Mobile Cloud Computing (MCC) has been proposed as a solution for mobile devices that need to carry out high-complexity computation with limited resources. Technically, with MCC, high-complexity computation tasks are offloaded from mobile devices to cloud servers. However, MCC does not work well for time-sensitive mobile applications due to the relatively long latency between mobile devices and cloud servers. Mobile Edge Computing (MEC), is expected to a latency optimized version of MCC. With MEC, edge servers, instead of cloud servers, are deployed at the edge of the network to provide offloading services to mobile devices. Since edge servers are much closer to mobile devices, the resulting latency is significantly lower than the cloud servers. Despite the advantages of MEC over MCC, edge servers are not as resource-abundant as cloud servers. Consequently, when many offloaded tasks arrive at an edge server, admission control needs to be in place to arrive at the best performance. In this thesis, we propose a Deep Reinforcement Learning (DRL) based admission control scheme, DAC, to maximize the system throughput of an edge server. The performance of DAC is thoroughly investigated via extensive simulations. Specifically, in our simulations, when varied offloaded tasks arrive at an edge server, an admission control scheme determines whether a task is admitted or rejected. Our experimental results indicate that DAC outperforms the existing admission control schemes for MEC in terms of system throughput and resource utilization.

Acknowledgements

At this moment, I want to give my greatest gratitude to my supervisor Dr. Qiang Ye for his professional guidance, valuable advice, and knowledge, which strongly benefited me and accelerated my research work. Moreover, I am also grateful to my colleagues Dr. Hui Huang, and Dr. Yuxuan Jiang, who spared their precious time to discuss and brainstorm with me. Their suggestions, support, warmheartedness and encouragement as well continuous helpful criticism benefited me extremely during this research.

Lastly, my deepest appreciation goes to my parents Mr. Kai Zhou and Mrs. Yan Hua and the rest of my family for their continuous care, support and encouragement. This research can not be accomplished without their unconditional love and affection. Thank you all.

Chapter 1

Introduction

1.1 Motivation

With the wide deployment of mobile devices, e.g. smartphones, the quantity and type of mobile applications have grown rapidly over the past years. Nowadays, many mobile applications, such as online games and virtual reality, involve high-complexity computation that requires a large number of hardware resources, which often exceeds the processing capacity of mobile devices [1, 2, 3, 4]. Even if mobile devices have sufficient resources to execute these applications, the high-complexity nature of these applications typically leads to a long execution time, which is not energy-friendly to battery-powered mobile devices.

Therefore, to meet the computation requirements of these mobile applications or reduce the energy consumption of mobile devices, Mobile Cloud Computing (MCC) was proposed as a solution to this computation and resource limitation problem. With MCC, high-complexity computation tasks are offloaded from mobile devices to cloud servers. Therefore, other than utilizing the network bandwidth and battery on offloading, the mobile devices do not need to allocate any hardware resources for offloaded tasks. Once cloud servers complete the offloaded tasks, the computation results are returned to mobile devices. As a result, the MCC replaces the complex, resource heavily used and long-time executed tasks with relatively simple, resource-friendly and short-time used offloading actions. Therefore, the MCC saves the mobile devices' hardware resources and extends their battery life, while these mobile devices are running the applications with complex functions and high-quality images.

To offload the high-complexity tasks to cloud servers, wireless technologies, such as LTE/5G[5, 6], are used to transfer data by mobile devices.The LTE 4G, as a high-speed wireless technology, has increased the data transmission speed up to 100 Mbps.

This transmission speed significantly deducts the delay in offloading tasks and returning computation results. Therefore, LTE 4G makes MEC finish offloading tasks before the deadline possible. The 5G technology further improves the bandwidth to 10 Gbps to further reduce the delay of the MEC. However, since 5G is using millimetre wave, which is difficult to maintain long-range propagation and low penetration loss, in physical layers. Therefore, the LTE 4G is more reliable in task offloading and the 5G can optimize the delay performance in MEC. However, due to the long distance between mobile devices and cloud servers, the lengthy latency between mobile devices and cloud servers tends to be intolerable to time-sensitive applications. To process the offloading tasks with lower latency, Mobile Edge Computing (MEC) is proposed to deploy less-powerful servers at the edge of the network and offload requests to the relatively close edge servers (instead of remote cloud servers).

Despite the obvious advantages of MEC over MCC, edge servers are not as resource-abundant as cloud servers. To be specific, with the benefits of hardware virtualization, the cloud servers are virtualized from the clusters of powerful servers. In this way, virtualization technology provides independence from the underlying hardware for services and operating systems [7]. Therefore, the cloud servers should not be considered as physical machines, whose configurations are static. The cloud servers can scale up and down their hardware capacity in need by invoking and releasing the hardware resource within the network of its cluster. Nevertheless, the edge servers, which are established as singleton with the base stations. There are no extra physical machines working as a cluster where the edge server can adjust its hardware capacity. Therefore, the edge servers normally can not enhance their hardware capacity by utilizing the spared hardware resources within a cluster. Furthermore, the cloud server clusters can increase their hardware capacity by including more physical machines into the clusters with virtualization administration software, while the clusters are providing the services. On the contrary, the edge server must replace the machine itself, when the edge server must be halted. As a result, the MCC can theoretically provide unlimited hardware support, but the MEC will finally run out of its hardware resources if the mobile devices' use cases keep expanding. Consequently, when many offloaded tasks arrive at an edge server, admission control needs to be in place to arrive at the edge server's best performance.

Several admission control schemes are developed to improve the system throughput of edge servers. However, most of the algorithms are focusing on optimizing the load balance and resource-saving according to the network topology. When it comes to the algorithms that work on a single edge server, the common mechanism is sorting the offloaded tasks by a developed cost equation to admit the tasks with a preference for lower resource requirements and lower execution period. Though admitting tasks with a priority can ensure the edge server accept the tasks with relatively low resource requirements and occupation time, the low-priority tasks can still be accepted as long as the edge server has sufficient resource. Therefore, even if the cost equation can assign the task running time an extremely large coefficient to maximize the cost of the task and to minimize the priority of the task, this task request can still be admitted and executed if the edge server's spare hardware resource meet its requirements. In other words, the sorting-based admission control policies can not successfully clear the resource-intensive requests, whose negative influence is mainly contributed by the executing time. Therefore, this thesis is devising an admission control policy, which can reject the resource-intensive tasks actively to spare hardware resources for more offloading tasks, especially when the edge server's hardware resources are largely allocated. Hence, the edge server increases its throughput of handling offloading tasks by ensuring the hardware resources for resource-intensive tasks, even if it has to abandon part resource-intensive tasks.

1.2 Overview of the Proposed Scheme

The objective of this thesis is to develop an admission control mechanism to increase the edge server's throughput. The proposed scheme utilizes Deep Reinforcement Learning (DRL) to solve the admission control problem in MEC. Specifically, we propose a novel admission control scheme for MEC, DRL-based Admission Control (DAC). With DAC, a DRL algorithm, Deep Deterministic Policy Gradient (DDPG) [8], is employed to determine whether an offloaded task should be admitted or rejected so that the system throughput of an edge server is maximized. Therefore, the high-priority tasks can be accepted, and the influence of low-priority tasks can be mitigated. Technically, DDPG is a DRL algorithm that is capable of learning in a high-dimensional policy space. By observing the request details of historical tasks and the impact of the tasks on available resources on edge servers, DDPG gradually learns the best admission policy for an edge server in MEC. Through extensive experiments, we found that DAC outperforms the existing admission control schemes for MEC in terms of system throughput.

In this thesis, we present a scheme that decides the admission of requests to optimize the performance of edge servers by accepting resource-friendly requests and rejecting the potentially resource-intensive requests. The process of this admission control scheme is implemented with the following steps:

(1) Develop a policy-based admission control system, which can accept and reject the incoming requests with an admission policy.

(2) Extract the parameters, such as resource usages and request requirements, as a vector that can describe the status of the system.

(3) Construct a neural network, which can take the system status vector as input and generate the admission policy.

(4) Further develop a reward equation that produces the feedback of the system for the model's learning.

(5) Train the neural network to pick a neural network that generates the admission policy increasing most performance.

(6) Evaluate and compare our scheme to existing solutions on the simulation system.

1.3 Thesis Outline

The rest of this thesis is organized as follows. The detailed background on admission control, MEC and DRL is presented in Chapter 2. It also illustrates the related research that has been processed. Some of research involves the existing algorithms and performance evaluation. At the end of this chapter, the problem under investigation is formulated. The proposed admission control scheme, DAC, is described in Chapter 3. Chapter 4 includes the detailed experimental results. Finally, our conclusions and future work are described in Chapter 5.

Chapter 2

Related Work

As introduced in the above sections, the scale of mobile computing scenarios has been inflated by the development of remote resource utilization. MCC and MEC, the typical techniques that provide powerful computation for mobile devices, are hot topics in increasing the performance of mobile computing. On the other hand, within a network, the maximum system throughput can be considered as the constraint of task offloading. Therefore, throughput maximization is a critical aspect to be researched for further performance improvement. When it comes to optimizing the performance of MEC, DRL is also a common methodology implemented by the researchers. Therefore, this chapter discusses and presents the topics and research that has been conducted in MEC, throughput maximization and DRL.

2.1 Mobile Edge Computing

The concept of MEC is extended from MCC. The MCC is utilizing a rich portfolio of services and applications from the data center or computation center [9]. With this methodology, the mobile devices' application can get rid of the constraints of the hardware capabilities and battery life.

In comparison with the MCC, the MEC's service provider is on the edge of the network [10]. To be specific, in mobile computing, the edge service provider normally refers to the base stations(BSs) and wireless access points(APs). The devices used to work as the transmitters in a specific network have become powerful service providers because their storage and computation capacities have improved the development of hardware and software. As Mao *et al.* proposed, the hardware of edge service providers is capable of processing tasks like machine learning and augmented reality [11]. Furthermore, since the service providers directly connect to the mobile devices, the latency of the services is significantly reduced. According to the recent research, Weisong claims that the delay of the edge computing is only about 169 ms, while the delay of the cloud computing is about 900ms [12]. Therefore, this proximity, which indicates a shorter distance between mobile devices and servers, contributes to a smaller latency for MEC to handling offloading tasks.

As for the disadvantage of the MEC to the MCC, the hardware capacity of the edge server is considered obviously less than the cloud server[12]. To be specific, the difference between edge servers' and cloud servers' hardware capacities is not only based on the physical hardware components but also based on the service deployment and technologies. The edge server is deployed on the base station, which is at the edge of the network. Meanwhile, the cloud server is deployed at the data center or computation center, where hardware components can be freely placed. Therefore, MEC's hardware capacity is weak compared to the MCC. Furthermore, because of the development of virtualization technology, the hardware capacity of the cloud servers gits rid of the physical space [13]. Moreover, the implementation of the auto-scaling allows the cloud server to increase its hardware capacity when it is running out of its current resources, without the change in the physical machine [14]. Therefore, regardless of the physical difference between MEC and MCC, the edge server is more possible to meets the problem of the resource constraints.

To increase the efficiency of implementing edge servers, the researchers propose the work in several different directions. Considering the performance and features of mobile devices, edge servers, and cloud servers, offloading scheduling is promised as an important aspect to optimize the efficiency of MEC. If the tasks are offloaded to the most suitable devices, the energy consumption, task latency, and completion rate are optimized [15, 16, 17, 18]. With these basic theories and approaches, further optimizations are developed with reinforcement learning and DRL [19, 20, 21, 22, 23].

Regardless of the optimizations on the offloading, if the edge service providers have sufficient resources, the more tasks are offloaded, the mobile devices have the higher efficiency to accomplish the tasks. Therefore, maximizing the system throughput is a reliable aspect to improve the performance of the MEC.

2.2 Throughput Maximization

As the previous section summarized, optimize the system throughput of MEC is a significant approach to improve the performance of MEC. Therefore this section describes the specific theories and methodologies that are applied in this MEC throughput maximization problem. In the aspect of mechanism, the major solutions for MEC to increase its throughput are categorized as physical-based, virtual-network-based, and admission-control-based. The specific topics and researches are discussed briefly below.

2.2.1 Routing and Physical Optimization

The feature of the wireless network defines the connection quality within the network is dependent on the distance between the nodes. Considering the mobility of mobile devices, optimizing the network routing and geometrical relationship to increase the MEC's throughput is also a direction of research.

In the research [24], Xiumei *et al.* devised an optimization algorithm, which includes all variables in the MEC. In this environment, the routing and offloading decisions are based on the status, such as connection quality, task queue backlog and energy consumption , of the base stations. Therefore, the better connection between mobile devices and bases stations, the more tasks and requests can be offloaded to the edge server. Therefore, Xiumei *et al.* optimizes the edge server's system throughput by improving the communication efficiency physically. In addition, since this research also involves the energy harvesting(EH) technique, optimization the throughput of energy transferring is also a target for optimization [25]. Xiumei *et al.* firstly utilize the perturbed Lyapunov optimization to get a routing matrix, which makes base stations admit the most number of offloaded tasks. Furthermore, the author accompanied the cost of the queue of backlog into the optimization of the routing matrix to improve the system throughput of the MEC.

On the other hand, [26] illustrates a solution that improves the system throughput by optimizing the geometrical relationship between nodes. In this research, Jie *et al.* introduced a case where functions of base stations are performed by the unmanned aerial vehicle (UAV). Therefore, adjusting the location of UAVs can modify the geometrical relationships between the nodes. As a result, the connection quality and communication efficiency can be improved by these modifications. Therefore, Jie *et al.* devised a deep Q learning based algorithm to supervise the natural environment and connection quality. With the observations from supervising, this algorithm produces a set of adjustment decisions for UAVs to update their locations. Consequently, the system throughput is successfully improved by this solution.

2.2.2 Network Virtualization

Network virtualization is a technique that enables the abstraction and shares the hardware and physical resources [27]. This technique also further devised two techniques, namely Network Function Virtualization (NFV) and Software Defined Network(SDN), with the emergence of MEC. Both two techniques flexibly administer the resources within the network without the constraints of the physical hardware [28]. Therefore, the NFV is considered a common paradigm in the future network communication and several throughput maximizations is developed based on NFV.

Yu *et al.* devised the provisioning mechanism in an NFV-enabled MEC system [29]. Then Yu *et al.* developed a cost-based algorithm to evaluate NFV's request multicasting mechanism. With the estimation of the costs, a multicast tree, whose subset of edges connecting the source node and destination, is constructed. This tree indicates this request can be admitted with the estimated cost. Finally, Yu *et al.* minimized the cost of every multicast tree to maximize the system throughput of this MEC system.

Similarly, Yi *et al.* also implemented the network virtualization to optimize the system throughput of the MEC [30]. The difference of this solution from Yu *et al.* is Yi *et al.* focusing on the administrations on the VNF but not multicasting. To be specific, Yi *et al.* formulate every offloaded task from the mobile devices into a set of small-size tasks, which can be processed by the service functions chains(SFCs). With these SFCs, the separated tasks can be handled by the VNF sequentially. Therefore, Yi *et al.* developed an algorithm that can estimate the correlation between the requesting node and processing node. Then this most correlated request will be processed by the node whose spare resource is sufficient but not larger than another node. In this way, this MEC system processes efficient requests with a high priority and fully utilizes the resources within the network. Consequently, the system throughput will be increased by admitting more efficient requests.

2.2.3 Admission Control

Admission control is the methodology for a single server to admit or reject incoming tasks. In the case of MEC, though, the edge servers have richer resources compared with the mobile devices. In comparison with the cloud servers, the capacity of the edge servers is not as sufficient. Even if the edge server is powerful enough for current use cases, it will finally run out of its hardware resources because of the growing number of mobile devices and the increasing diversity of functions provided by the mobile devices. Therefore, providing more services with constrained hardware resources is an important optimization problem for the edge server. In this case, admitting the task requests with a specific strategy is helpful for edge servers to process more tasks with their resource constraints. In recent years, there are several solutions that involve the specific admission control policy that improves MEC's system throughput.

Online Batch Algorithm [31]

Xia *et al.* proposed a cost model to estimate the influence of every request on the system throughput. Theoretically, Xia *et al.* formulates the throughput maximization problem into a bin packing problem. Every hardware resource of the edge server can be considered as empty bins with specific capacity. Therefore, Xia *et al.* claims that the edge server can perform better by admitting more requests with fewer resources requirements and shorter occupation time, just like admitting smaller items in the bin packing problem.

Xia *et al.* firstly developed an equation to estimate the cost of resources. To be specific, every resource is set with a basic value, which can be adjusted based on the capacity of the server. To be specific, if some hardware resources, such as memory, are easily running out, the server can configure a larger value for the admission control to prefer to admit tasks with fewer memory requirements. Then the cost of task resources is generated by raising this basic value to the power of the ratios of resource utilization and task resource requirements. Finally, the cost of the task is the product of the cost of task resources and the task's running time.

On the other hand, Xia *et al.* also devised a threshold system, which can check whether the request is obviously resource-intensive. Therefore, after assigning every offloaded task with their estimated cost to the edge server, the incoming requests need to be filtered if their hardware requirements exceed the server's available resource or their estimated cost exceeds the predefined threshold.

When filtering is finished, the edge server sorts the incoming tasks by their costs. The less cost is estimated, the earlier this task is checked by the edge server. Therefore, considering the decrease of available resources caused by the admission, the lower cost of the task is estimated, the higher probability that the task can be admitted. Every checked task is admitted if the edge server has sufficient available hardware resources that meet the task's resource requirements. Otherwise, this task is rejected because it can not be processed by the edge server. Theoretically, admitting the requests whose cost is less than the others, can spare more available resources for other tasks. Moreover, since the cost involves the influence of the running time, over the long term, more space for the cost of the future request is spared. Therefore, in the case of admitting tasks, the admitted requests utilize fewer resources and run for less time. Therefore, this algorithm minimizes the requests' average influence on the system throughput. As a result, in a relatively long time interval, the edge server's system throughput is increased by this algorithm.

Maximum Efficiency First Ordered (MEFO) [32]

Hu *et al*. proposed a similar solution. However, this admission control algorithm is applied to a multi-server environment. Furthermore, the ratio of the available resource is not a factor to calculate the cost of the task, but a important factor to guide the offloaded task to identify suitable server. In this research, the ratios of the required resources to the edge server's spare resources are designed for mobile devices to select edge servers. To be specific, the smaller ratios are, the more suitable is this edge server for offloaded tasks. In this way, according to the requirements of the task, the task can be offloaded to a more suitable server, whose ratio of the available hardware resources decreases smallest. To be specific, the requests, which occupy more memory resources for their tasks, are sent to the servers, whose memory resources are richer than the other resources. Moreover, if more than one edge servers meet the previous requirement, the task is offloaded to the edge server, whose overall available resource is more. To simplify this edge server selection, Hu *et al*. developed a matrix that is used to calculate the tasks' scores on different servers.

As for the admission control algorithm working on the single edge server, the solution proposed by Hu *et al*. also implemented a similar logic. The difference is, Hu *et al*. devise a different cost calculation. Unlike the ONLINE-BATCH algorithm, the MEFO will not estimate the tasks' resource costs following exponential growth. The MEFO just sums up the ratios of required resources to the edge server's available resources as the task's resource cost. Then the MEFO also gets the weight of the task by timing the cost of task resource requirements with its execution time. Finally, the tasks will be reordered by their weights and be checked by the edge server one by one. The edge server's performance is improved by this algorithm like the ONLINE-BATCH algorithm verified.

2.3 Reinforcement Learning

Reinforcement learning (RL) is an area of machine learning. However, this machine learning is not based on the neural network. The basic function of RL is maximizing the notion of the cumulative reward[33]. To accumulate the reward, the learning model needs to estimate the reward, which can be acquired at the current state with a specific action. With a well-trained RL model, the actor can always get the correct estimation about the final reward. Therefore, at every stage, the actor just needs to process the action whose estimation of the reward is the largest to get the final reward. A sequence of combinations of the state and actions is working as a Markov chain. Therefore, the basic idea for RL's reward estimation is times the final reward with the probability of finally getting the reward, which is calculated with the Markov decision process.

2.3.1 Q-learning

Theoretically, if the actor knows all of the details of the states, this problem normally can be reformulated into a shortest path problem or minimum spanning tree problem. Moreover, in most cases, there is no specific probability that can be concluded for the actions at a specific state. Therefore, the RL may not solve the problem more efficiently. To get rid of the prerequisite problem of knowing all environment states and the probabilities of all actions, Q-learning is proposed. The major contribution of the Q-learning is involving the Bellman equation in the final reward estimation[34]. With the Bellman equation, a Q value, which represents the estimation of the final reward, can be calculated. According to the Bellman equation, the Q value at the current state is a discount of the max Q-value of the states, which can be achieved from the current state. Therefore, even if the actor does not know the environment in the first place, the actor can also explore the environment randomly. Once the actor gets the reward, the Q value will be back-propagated along the path, which is explored by the actor. After several iterations, a well-trained Q-learning agent can be utilized to optimize the solution to handle the current environment.

2.3.2 Double Q-learning

Regardless of the performance and efficiency of Q-learning, Q-learning's learning procedure can be easily biased. As introduced before, Q-learning will calculate the Q-value along with the path, which the actor takes to reach the reward. However, once some of the states are calculated and marked with the Q value, in the further learning procedure, the actor may be easily attracted back to this old path, even if there is another shorter path to the reward. Therefore, the learning procedure of the Q-learning may finally make the Q-learning only achieve a suboptimal solution. To make Q-learning more possibly get the optimal solution, Double Q-learning is proposed. The idea of Double Q-learning is training multiple Q-learning agents at the same time [35]. In this way, two biased Q-learning models are trained. However, combining the outcomes of two Q-learning models, a less biased Q-learning model can be picked out. Repeating the separated training and outcome combining until the difference between two Q-learning models is close to zero, an unbiased Q-learning model is trained.

2.4 Deep Reinforcement Learning

Despite the outstanding performance of RL in maximizing the reward within a specific environment, RL's mechanism limits its use cases. No matter what RL methods, all trained models should have a map for all available states and their final reward estimation. If the size of environment states is large enough, the RL will be progressed with extremely low efficiency. In the worst-case scenario, RL may fail to reach the reward until it is used out of the memory or storage for its model. The function of the outcome of RL is combining the states with their estimation of reward. Therefore, if there is a computation method, which can calculate the estimation of reward with the state, the storage space for RL can be released. Since the neural network can compute the complex input to get high dimension output, which is close to the logic binding the state with its estimation, the concept of the DRL is developed [36].

2.4.1 Deep Q-Learning

Deep Q-learning, is the most naive combination of the Deep Neural Network(DNN) with the RL. As the concept of the DRL, deep Q-learning is trying to use the neural network to mimic the procedure of Q-learning. When the deep Q-learning explores the environment, the steps of getting a reward and calculate the Q value are the same as the Q-learning. After the calculation of the Q values, the states and Q values are used as the input and the target of the neural network. However, the DRL model, which is trained with this learning procedure, is unstable. Normally, these DRL models can not get the correct Q values with the state, since the states and the Q-values do not have a linear relationship. Therefore, a technique, named experience replay, is implemented to remove the correlations in the state sequence and smooths changes in the data distribution [37]. And the replay buffer becomes the necessary component for the DRL.

2.4.2 DRL with Actor-Critic Methods

Though deep Q-learning makes RL gets rid of the limitation of the state space, the size of the actions still influences the performance of the DRL. According to the procedure of the RL, even if deep Q-learning is implemented, the learning model progress slow in summarizing the estimations, which is caused by the possible actions. Furthermore, similar to the bias of the Q-learning, the deep Q-learning also potentially learns a suboptimal model or never reaches the reward. Similar to the solution to tackle the problem of state space, the neural network is the solution for the large action space. Therefore, the relation from state and action to the reward needs to be learned by the neural network [38]. Moreover, to simplify the procedure of picking an optimal solution, a neural network, which is used to infer the suitable action from the state, is implemented. This solution is the actor-critic method, where the actor network is trained to pick the optimal action and the critic network is trained to estimate the Q value.

Furthermore, the critic network is not only capable of predicting the Q value or estimation of the reward. In the implementation of the Advantage Actor-Critic (A2C) algorithm, the critic network is used to predict the baseline of the estimation at the current state [39]. In this way, the critic network can predict the baseline value only with the state. The difference between the Q value, which is calculated by the environment, and the baseline value is called advantage. Basically, the advantage gives the direction of the optimization. If the advantage is positive, which indicates that the current A2C model may underestimate the environment, the actor network and the critic network are optimized towards the environment to improve the baseline for further improvement. Otherwise, the current A2C model may overestimate the environment, the actor network and the critic network are optimized backwards the environment for more precise predictions and improve the policy. Therefore, the A2C is not only mimicking the function of the Q-learning algorithm but also optimizing the learning procedure.

In the research, the algorithm Volodymyr *et al.* introduce is the Asynchronous advantage Actor-Critic algorithm (A3C) [39]. This algorithm's mechanism is the same as the A2C, but A3C implements a multi-agent method. In this way, the A3C models can be trained separately. Once training is done, all A3C networks are compared with each other. The models that perform better can be used to update the global network. According to the experience of Double Q-learning, the A3C should have better and more stable performance. Unlike the learning procedure of Q-learning, which updates the Q value to the target directly, the neural network updates its variables with a learning rate for the precision of learning. Therefore, the A3C algorithm does not obviously outperform the A2C, but it proposed a framework for the multi-agent neural network.

2.5 MEC with DRL

As introduced before, the edge server is powerful enough to process the machine learning task. This use case brings a new topic, which is known as edge intelligence. This topic refers to the implementation of deep learning within the MEC. Normally, these implementations will only provide the machine learning services for mobile devices. In [40], the author developed an Internet of Things (IoT)-based energy management scheme to schedule the energy consumption within the smart cities. To be specific, the author utilizes the DRL to observe the energy consumption demands and produce actions to schedule the energy consumption. Since the value of the saved energy consumption is the reward for DRL, this DRL provides a successful energy management service. In the industrial aspect, the concept of Industry Internet of Things(IIoT) is raised. The techniques of MEC and machine learning become a methodology to improve productivity by optimizing configurations of the production line [41]. Meanwhile, machine learning is also applied in quality control to lower the maintenance costs [42].

These machine-learning-based services can not only serve mobile devices, but also be implemented by the edge servers themselves. MEC can utilize the inner machine learning services to optimize the resource utilization and throughput maximization [43]. This kind of edge intelligence is called intelligent edge. In the previous discussion, [26] implements machine learning to optimize the positions of the UAV.

In [44], the author noticed that methodology that handles the network delay in formulating the existing problem into a Mix Integer Nonlinear Programming (MINLP) problem. Then the MEC leverages the Lyapunov optimization method to solve the formulated problem. However, the time complexity of Lyapunov optimization is large. Zhaolong *et al.* utilizes the machine learning to replace the calculation in Lyapunov optimization. Therefore the trained model can imitate the decisions by observing the previous input.

Other than the latency reduction, the machine learning services are also applied to network security. The most common function is to classify the exceptional functions and abnormal interactions within the network. To be specific, in some of the research, neural networks are trained to learn the features of network interactions to detect spoof attacks [45, 46]. Theoretically, rejecting or blocking these interactions, when the neural network detect them, can effectively prevent the network from the poisonous attack. Furthermore, the activities of processes and files on the edge server can be monitored by the machine learning services [47]. Therefore, once the deep learning services detect any of the edge servers having abnormal files, the administrator can protect this network by expel the suspect edge servers from the trust list until they fix files.

Chapter 3

DAC: DRL-based Admission Control for MEC

3.1 Problem Formulation

In this section, we formulate the throughput maximization problem under investigation. Specifically, we consider a 2-tier MEC system in our research. The architecture of the MEC system is illustrated in Fig. 3.1. In this system, there are a number of mobile devices, which attempt to offload their high-complexity computation tasks to an edge server that is located at a base station. After receiving the tasks from varied mobile devices, the edge server can either accept or reject the request. A list of the key notations used in this paper are listed in Table 3.1. The details of the notations will be presented in the rest of this paper.

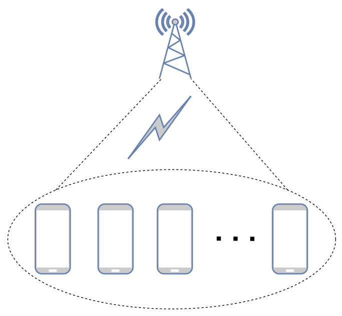


Figure 3.1: MEC Architecture

In an MEC system, an edge server typically receives a set of offloading requests, which consume the resources of the edge server. In our research, this set of requests from mobile devices are denoted as $Q = \{q_1, q_2, \cdots\}$. Considering the limited hardware resources of mobile devices, these requests will utilize the hardware resources for amount of time on the edge server to save their resources usage and batteries' lives. Therefore, every task request from mobile devices should be patched with its requirements of hardware resources and its running time. To simplify the description of requests arrival, a specific time interval can be considered as a time slot. In this

Table 3.1: Key Notations

Notation	**Description**
Q	Request set
q	Request
q_i	i-th request in request set Q
τ_{max}, τ_{min}	Maximum and minimum execution time
τ_q	Execution time for request q
E	Condition of edge server
H	Occupied system resource
A	Available system resource
a_t	Admission decision

way, the requests, which arrive in the edge server within this time interval, can be considered as arrive in the edge server at this time slot. With the concept of time slot, all admission policies, which will decide the acceptance and rejection of requests, are generated at the beginning of every time slot t.

To be more specific, there are K different hardware resources in this system. For edge server, C_k can be denoted as the capacity of resource k on the edge server. In this case, k is an integer in the range of 1 to K. Moreover, the relative properties, the amount of occupied resources and available resources, can be denoted as H_k and A_k. For a specific time slot t, these properties can be grouped into two vectors, $H(t) =< H_1(t), \cdots, H_K(t) >$ and $A(t) =< A_1(t), \cdots, A_K(t) >$. At time slot t, all resources should satisfy Eq. (3.1). Since a request will occupy the hardware resources for several time slots, the request should be described with the resources and the occupation time. Therefore the request in the request set $Q(t)$, which arrives at edge server at time slot t, is denoted by $q_i(t) =< q_{i,1}(t), \cdots, q_{i,K}(t); \tau_i >$. In this manner, $q_{i,k}(t)$ represents the amount of resource k that is occupied by request i arriving at time slot t; and τ_{q_i} represents the occupation time of request q_i.

$$A_k(t) = C_k - H_k(t) \tag{3.1}$$

For the MEC system under investigation, system throughput is defined as the number of completed requests during a specific period T. Consequently, throughput maximization is about admitting as many requests as possible within the time window T. During each time slot, a number of requests arrive at an edge server. These requests, which are filtered by algorithms, can be formally admitted if the edge server can provide sufficient hardware resources. Otherwise, even if some of requests pass the filtration of algorithms, they will be rejected and not be counted as the throughput of edge server. The admitted requests will be executed by the edge server, at the same time, the rejected requests will be executed by mobile devices or resubmitted to other edge servers or cloud servers. Once the requests are admitted and executed by the edge server, these requests will start running and be assigned with hardware resources they require. These resources will become exclusive for the request claims them until this request finish its work and release these resources.

3.2 Details of DAC

In our research, we attempt to utilize deep reinforcement learning to arrive at the best admission policy in order to maximize the system throughput in an MEC system. Though the DRL is implemented in the optimization in the routing and server selection of MEC, this is the first attempt in maximizing the throughput of single server of the MEC. Specifically, we use the DDPG algorithm to learn the appropriate admission policy by observing the request details of historical tasks and the impact of the tasks on available resources on edge servers. The workflow of the proposed admission control scheme, DAC, is illustrated in Fig 3.2. As the figure states, the DRL will produce admission policy with the input, which represents the status of the edge server and incoming requests. These admission policies firstly ensure the admission for more resource-friendly tasks. As for other tasks, when the edge server is busy and heavily utilized, DAC can reject the resource-intensive requests, which contain the heavy tasks separated from games, VR or AR, to spare available resources for more tasks. Otherwise, when the edge server has large amount of spare resources, the resource-intensive requests can still be admitted by the DAC. The details of DAC are presented in the rest of this section.

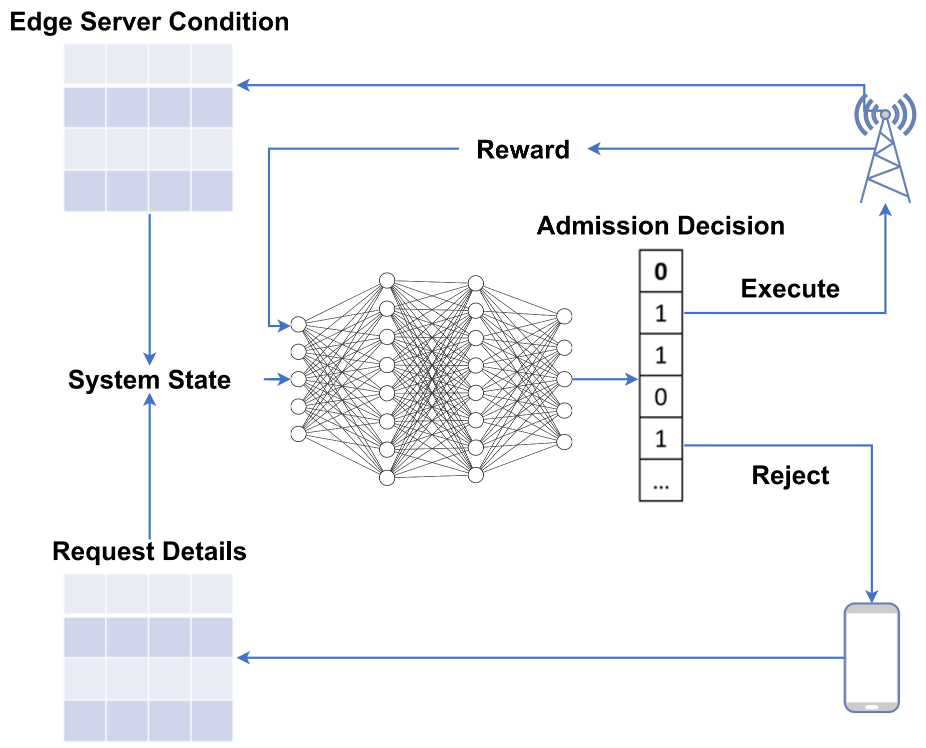


Figure 3.2: Workflow of DAC

Overview of DDPG

DDPG is a DRL algorithm based on the actor-critic structure that was proposed to deal with optimization problems by providing high-dimension action or policy [8]. The details of DDPG are summarized in Alg. 1. Up to now, DDPG has been the methodology for the several solutions to communication applications [43, 48]. As for the details of the DDPG, to be specific, this learning model consist of four neural networks, namely actor network, actor target network, critic network, and citric target network. In the further discussion, these networks are denoted with $\pi_{\theta^{\mu}}$, $\pi_{\theta^{\mu'}}$, $\pi_{\theta^{Q}}$, and $\pi_{\theta^{Q'}}$, respectively.

The actor networks are performing decision generating with the observations. The mechanism of the actor network is mimicking an actor observing the environment and status of the system, then produce the policy for the system. Therefore, this network takes the observations or system states as input and calculates the policies as output.

Meanwhile, the critic networks are designed to evaluating the policies that are generated by actor networks. The basic evaluating mechanism is the critic networks take the combination of the observations and policies as input and takes the reward value as the target value. Therefore, the reward value is the evaluation result for the policies, which are taken based on the observations. While the critic network $\pi_{\theta Q}$ is processing learning procedures with these data, the underlying logic of evaluating the policies, which are applied to the observed system, can be learned by the critic network. After convergence, the critic network can predict the reward for the input, which represents the different states of systems applying different policies. With the help of the critic network, the actor network can be guided to produce the policies, which can make the system feedback with higher reward value. On the other hand, the target networks are the networks $\pi_{\theta \mu'}$ and $\pi_{\theta Q'}$ that stabilize the learning process their relative networks by reducing the value change of the network.

Algorithm 1 DDPG

```
1: Initialize actor and actor target network $\theta^{\mu}, \theta^{\mu'}$;
2: Initialize actor and actor target network $\theta^{Q}, \theta^{Q'}$;
3: Initialize replay buffer $\mathcal{M}$;
   *LOOP Process*
4: **for** episode t in 1,2,3,... **do**
5:     **for** t=1,2,3, ... **do**
6:         Observe state $s_t$;
7:         Generate the admission policy $a_t = \pi_{\theta^{\mu}}(s_t) + \epsilon$ and calculate reward $r_t$ and
           next state $s_{t+1}$;
8:         Store transition $(s_t, a_t, s_{t+1}, r_t)$ into $\mathcal{M}$;
9:         Pick sample mini-batch of transitions from buffer $\mathcal{M}$ and reorganize them
           into matrices: $s_{t_n}, a_{t_n}, s_{t+1_n}, r_{t_n}$;
10:        Smooth policy action $a_t : \widetilde{a_t} \leftarrow \pi_{\theta^{\mu'}}(s_{t_n}) + \epsilon, \epsilon \sim clip(\mathcal{N}(0, \tilde{\sigma}), -c, c)$;
11:        Loss function for critics is formulated as:
           $y_n \leftarrow r_n + \gamma \cdot \pi_{\theta^{Q'}}(s_{t+1_n}, \widetilde{a_{t_n}})$;
12:        Update Critic network:
           $\theta^{Q} \leftarrow min_{\theta^{Q}} N^{-1} \sum (y_n - \pi_{\theta^{Q}}(s_{t_n}, a_{t_n}))$;
13:        **if** $(t \bmod \delta = 0)$ **then**
14:            Update target network: $\nabla_{\theta^{\mu}} J(\theta^{\mu}) =$
               $N \sum \nabla_{a_{t_n}} \pi_{\theta^{Q}}(s_{t_n}, a_{t_n})|_{a_n = \pi_{\theta_{\mu}}(b_{t_n})} \nabla_{\theta^{\mu}}(s_{t_n})$;
15:            Update target networks
               $\theta^{\mu'} \leftarrow \nu\theta^{\mu} + (1-\nu)\theta^{\mu'}$;
               $\theta^{Q'} \leftarrow \nu\theta^{Q} + (1-\nu)\theta^{Q'}$;
16:        **end if**
17:    **end for**
18:    Continue till convergence;
19: **end for**
```

System Observation

At the t time slot, DAC observes edge server's state s_t, which involves the resource occupation and details about the incoming requests. As the system model described, the requests can be illustrated as a vector like $q_i =< q_{i,1}, q_{i,2}, q_{i,3}, q_{i,4} >$. In this case, the numbers from 1 to 4 refer to CPU, memory, disk storage and bandwidth, respectively. Therefore, for all $q_i(t) \in Q(t)$, can be combined and reformatted into a long vector to represent the observation of requests, which arrives at time slot t. Therefore, the details of requests set, whose size is at most 10, can be explained with $Q(t) =< q_{0,1}(t), q_{0,2}(t), \cdots, q_{1,3}(t), q_{1,4}(t), \cdots, q_{9,4}(t) >$.

The conditions of the system can be also illustrated in the same way. In the case of admitting request by available resources, the ratio of hardware resource occupation can be reasonable statistics to describe the situation of the edge server. Therefore, the conditions of the system can be described with $E_t =< \frac{H_1(t)}{C_1}, \frac{H_2(t)}{C_2}, \frac{H_3(t)}{C_3}, \frac{H_4(t)}{C_4} >$. Therefore, the state of the whole system can be $< q_{0,1}(t), q_{0,2}(t), q_{0,3}(t), \cdots, \frac{H_3(t)}{C_3}, \frac{H_4(t)}{C_4} >$. Furthermore, since the number of the requests arrives at every time slots are randomly generated, if the number of requests is smaller than 10, the $Q(t)$ will fill 0 to represent the non-existing requests.

3.2.1 System Action

DAC is designed to provide admission control policies for the edge server. Therefore, at the beginning of every time slot, it should determine whether every incoming request should be admitted or rejected by observing the edge server and incoming requests. As Alg. 1 illustrates, the final action a_t corresponds to the output of the actor network π_{θ^μ} with a randomly generated exploration noise ϵ. In this case, this output can be denoted with a_t and be calculated using Eq. (3.2):

$$a_t = \pi_{\theta^\mu}(s_t) + \epsilon \tag{3.2}$$

3.2.2 System Condition Update

Once a_t is available, the set of admitted requests will be updated using Alg. 2. And requests in the set will be processed by the edge server at the same time. At the end of the current time slot, some of the admitted requests will be completed. These requests, whose tasks are accomplished, should be removed from the admitted request set. At the same time, the resources, which are occupied by these requests, should be released. By releasing the occupied resources, the available resource vector should be increased accordingly. Alg. 3 includes the updating actions that need to be carried out. Finally, the ratios of resources occupations will be calculated again and denoted with E_{t+1}, which indicates the edge server's system status at the next time slot.

Algorithm 2 Admitted Request Set Update

Input: Request $q_i(t)$, admission decision $a_i(t)$, $H(t) = \{H_1(t), \cdots, H_K(t)\}$, $A(t) = \{A_1(t), \cdots, A_K(t)\}$

Output: Updated admitted request set $U(t)$

```
1: if $a_i(t) = 0$ then
2:     Reject $q_i(t)$;
3: else
4:     for resource $k = 1, 2, \cdots, K$ do
5:         if $r_{i,k} > A_k(t)$ then
6:             reject $q_i(t)$;
7:         end if
8:     end for
9:     for resource $k = 1, 2, \cdots, K$ do
10:        $A_k(t) = A_k(t) - q_{i,k}$;
11:        $H_k(t) = H_k(t) + q_{i,k}$;
12:    end for
13:    $U(t) \leftarrow U(t) + \{q_i\}$;
14: end if
15: return $U(t)$
```

Algorithm 3 System Update at the End of a Time Slot

```
Input: $U(t)$,$H(t) = \{H_1(t), \cdots, H_K(t)\}$,$A(t) = \{A_1(t), \cdots, A_K(t)\}$
Output: Admitted request set $U(t+1)$, $A(t+1)$,$H(t+1)$
1: $U(t+1) \leftarrow U(t)$;
2: $A_k(t+1) = A_k(t)$;
3: $H_k(t+1) = H_k(t)$;
4: for $q \in U$ do
5:     $\tau_q = \tau_q - 1$;
6:     if $\tau_q = 0$ then
7:         for resource $k = 1, 2, \cdots, K$ do
8:             $A_k(t+1) = A_k(t+1) + q_k$;
9:             $H_k(t+1) = H_k(t+1) - q_k$;
10:            $U(t+1) \leftarrow U(t+1) - \{q\}$;
11:        end for
12:    end if
13: end for
14: return $U(t+1)$, $A(t+1)$,$H(t+1)$;
```

3.2.3 Reward Function

The objective of DAC is to maximize the system throughput of an edge server while satisfying the hardware capacity of the edge server. To achieve this objective, when every accepted request meets the edge server's available resource, this admission control method will accumulate the reward value. This reward consists of three parts, namely the marks for throughput, execution time, and resource requirements. The mark for throughput is straightforward. If edge server accepts one request, its throughput will be increased by 1. Therefore, the mark for throughput is also a constant with value 1. As for the marks for execution time and resource requirements, the less execution time, and resources the request claims, the larger throughput this edge server is expected to have. Moreover, to normalize the influence of marks of different aspects, all marks are limited in the range of [0,1]. Therefore, denoting marks of execution time and resource requirements with ρ_t and ρ_r respectively. Hence, the calculation of marks and the rewards at time slot t is illustrated as follows:

$$\rho_t(q_i) = \frac{\tau_{min}}{\tau_{q_i}} \tag{3.3}$$

$$\rho_r(q_i) = 1 - \frac{\sum_{j\in\{1,\cdots,K\}} \frac{q_{i,j}}{C_j}}{K} \tag{3.4}$$

$$r_t = \sum_{\forall executed\ q_i(t)} \rho_t(q_i) + 1 + \rho_r(q_i) \tag{3.5}$$

The details of DAC are summarized in Alg. 4. To be specific, the state s_t is generated according to the ratios of resource occupations and details of the income requests. Next, if t is not zero, the transition $(s_{t-1}, a_{t-1}, s_t, r_{t-1})$ is stored into the buffer $\mathcal{M}$ for actor and critic networks' further learning. After this, DDPG's actor network produces the admission decision a_t with s_t. With the admission decision a_t, the edge server will execute the requests according to Alg. 2. Meanwhile, the reward will be calculated with Eq. (3.5). At the end of this time slot t, every running request's execution time will be reduced by 1. Those requests, whose execution time become 0, will be removed from edge server and release the resource they are occupying according to Alg. 3. Thereafter, the DDPG will update its networks. DAC continues until the stop condition is satisfied.

Algorithm 4 DAC: DDPG-based Admission Control

```
Input: A set of requests $Q(t)$, condition of edge server $E_t$
 1: for Time slot $t$,$t = 1, 2, 3 \cdots$ do
 2:     Observe the condition of edge server and details of requests:$s_t = (Q(t), E_t)$;
 3:     Store transition $(s_{t-1}, a_{t-1}, s_t, r_{t-1})$ to buffer $\mathcal{M}$
 4:     Produce admission policy $a_t$ with actor network $\pi_{\theta^\mu}$
 5:     Process admission using Alg. 2
 6:     Calculate $r_t$ with executed requests with Eq. (3.5)
 7:     Update edge server status using Alg. 3
 8:     Update DDPG networks using Alg. 1
 9:     Continue till convergence
10: end for
```

Chapter 4

Experimental Results

This chapter illustrates the context of the proposed solution and the results of simulation experiments. The primary goal of experiments is to verify the performance of the edge server processing the admission control with DAC. Specifically, the performance of DAC is compared to that of two existing admission control algorithms, which are designed to maximize the system throughput in MEC. Both algorithms are cost-based admission control schemes, which reduce the load or cost of edge servers in a long term to maximize the system throughput. The first algorithm, which is called ONLINE-BATCH [31], calculates the cost of a request, which is based on the average resource requirement and occupation time. The second algorithm, Most Efficiency First Offloading (MEFO) [32], employs a cost based on resource requirements and occupation time. In our research, as a baseline algorithm, the First-Come-First-Served algorithm is included in the comparison study. The experiments' results proved the proposed scheme make the edge perform better than other two algorithms.

4.1 Experiment Configuration

This section presents the experiment environment where DAC and counter solutions will be processed. The simulation environment consists of one edge server and a set of mobile devices that may send task requests to this edge server for processing as described in Fig. 3.1. Considering about the general use cases, this edge server provides the resources namely, CPU, memory, and storage. Their capacities are 2.99 GHz, 32 GB and 4 TB respectively. To be specific, the capacity of a single core of CPU is set with its computation frequency. Since the common modern server CPUs having multiple cores, we set the CPU in experiment is a CPU with 8 cores, whose max computation frequency is 2.99 GHz. Other than the resources for request tasks, this simulation also need to involve the network resource, which will be occupied by transmitting the requests and relative data. Since the edge server is set up with the base stations, the maximum speed of network transmitting is limited by the bandwidth of the edge server. Therefore, assuming the edge server is set up with the 5G technology ,the capacity of the bandwidth for wireless access is set to 10 Gbps [6, 49]. If this edge server is deployed at the base station with the LTE 4G, the configuration of the bandwidth should be modified. The other differences between LTE 4G and 5G will not influence the DAC's admission strategy. Furthermore, we assume the time slots refers to 10 seconds and the system monitoring time period is $T = 100$, which means every 100 time slots will be considered as an episode to check the system throughput. Meanwhile, the number of the request that arrives in the edge server is randomly generated within the range of [1,10]. For every request's specific resource requirements will be randomly generated within the range of resource in Table 4.1. In this table, the maximum occupation period τ_{max} is set with 20. Moreover, the range of CPU requirement is based on the proportion of the computation resource is occupied. In the specific calculation, the CPU's utilization can be recognized as the improvement of CPU's overall frequency. Therefore the requirement of CPU can also be roughly presented with *[0.15,0.3] GHz*.

Table 4.1: Request Specification

Type	**Range**
CPU (Proportion)	*[5%,10%]*
Memory (MB)	*[1000,2000]*
Storage (GB)	*[10,50]*
Bandwidth (Mbps)	*[50,100]*
Maximum Occupation Period (Time Slots)	*20*

Other than the proposed algorithm, the ONLINE-BATCH, MEFO, and FCFS are also involved in this experiment. To be specific, in the case of FCFS, the incoming requests are not processed specially. The edge server will keep admitting requests when it has sufficient resources. As for the ONLINE-BATCH and MEFO, the edge server will first remove the requests, whose resource requirements exceed the edge server's spare hardware resources. After filtering, the cost of every request is calculated according to the cost equation of ONLINE-BATCH or MEFO. Afterward, the rest of the tasks are sorted by their costs. At last, the edge server tried to admit these sorted tasks one by one, if it has enough resources.

To qualify the performance of the admission control algorithms, two evaluation metrics are used in our research. Moreover, to ensure all algorithms' performances are reliable, all experiments are processed for 2000 episodes. Furthermore, to explore the robustness of the all algorithms, we run more experiments whose requests having longer max resource occupation period.

- *System Throughput*: It indicates the number of finished requests within a period. Increasing this number ensure the more request can be processed by edge server.
- *Resource Utilization*: It indicates the efficiency of the algorithm to utilize the resource facing a shortage. To be specific, it can also indicates the performance of saving resources and leave space for incoming requests.

4.2 Experimental Results in Baseline Scenario

In this section, we compare DDPG-based admission control with the existing methods in the aspect of system throughput. The performance curves of these algorithms are plotted in Fig. 4.1. From this graph, we unsurprisingly notice the line of FCFS, which indicates the edge server with ordinary admission control policy, makes the edge server admit the fewest requests. Therefore, take the line of FCFS as a baseline, the other three lines can present how well these admission control methods optimize the edge server's system throughput.

On the other hand, the proposed algorithm outperforms all of its counterparts, MEFO, FCFS, and ONLINE-BATCH. To be specific, the performance of the proposed algorithm is about 60%, 33%, and 8% higher than the throughputs of the admission controls with FCFS, ONLINE-BATCH and MEFO respectively. As expected, the

DAC's strategies help the edge server successfully accept the most requests. And the DAC optimized the MEC's system throughput most efficiently.

In our theory, this advantage of the proposed algorithm is because the cost estimations of MEFO and ONLINE-BATCH, are focusing on the admitting less-cost request at the current time slot t while the DDPG-based admission control will reject some requests, which may decrease the system throughput in a long-term vision. Though both ONLINEBATCH and MEFO involves the occupation time as a factor of cost, they can not reject these requests if they are relatively expensive but still meet the edge server's available resource.

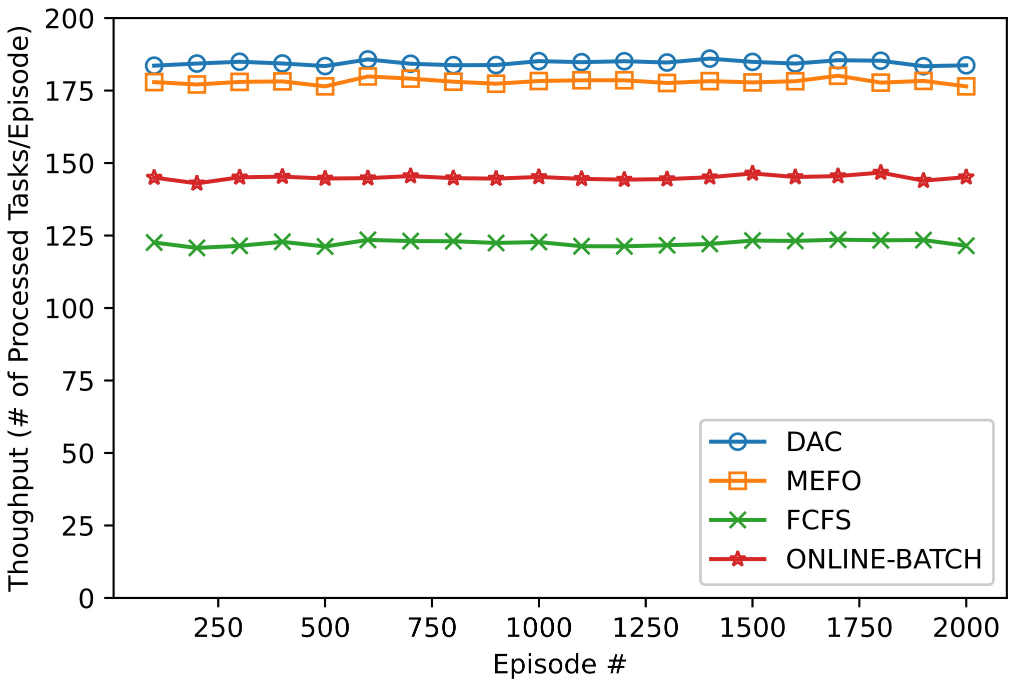


Figure 4.1: System Throughput with $\tau_{max} = 20$

As for the resource utilization, the performance of four algorithms can be explained with Fig. 4.2a, Fig. 4.2b,4.2c, and 4.2d. Observing four figures, the usages of storage and bandwidth are the smallest. All four admission control mechanisms only use around 10% of storage and bandwidth resources. This indicates most of these two system resources are not utilized. For incoming requests, storage and network bandwidth are always abundant in the edge server. Therefore, the requests acquiring the edge server's hardware usage are not possible to be rejected because of the shortage of either storage or network bandwidth. Moreover, differences in either storage or bandwidth utilization between the four algorithms are so slight that the advantage of DAC's resource allocating can not be firmly supported.

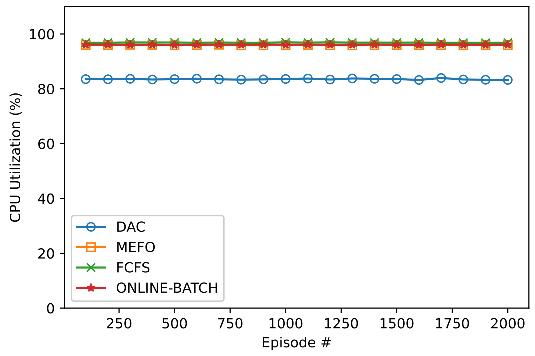
When it comes to the comparisons among four algorithms, though the DAC has the largest throughput, its resource utilization is the smallest. Moreover, other than ONLINE-BATCH, all admission control algorithms' performances are that the less the resource occupied, the higher throughput the edge server has. To be specific, in Fig. 4.2b, the edge server applying the proposed algorithm only occupied about 50% of the whole memory resource, which is roughly 70% of the resource utilization of ONLINE-BATCH.

Observing Fig. 4.2a, the edge server can be considered as suffering a shortage of CPU resources. Even if all of FCFS, MEFO, and ONLINE BATCH run out of CPU resources, the DAC's average CPU usage is still holding at 81Despite the less CPU utilization, the edge server with DAC accomplishes more task requests than the edge servers with other algorithms. This phenomenon indicates, DAC can save more system resources, which can be provided for admitting more incoming requests, than all other admission controls. Combining the line graph of memory utilization and system throughput, we can assume the DAC has a better performance in the resource-saving feature.

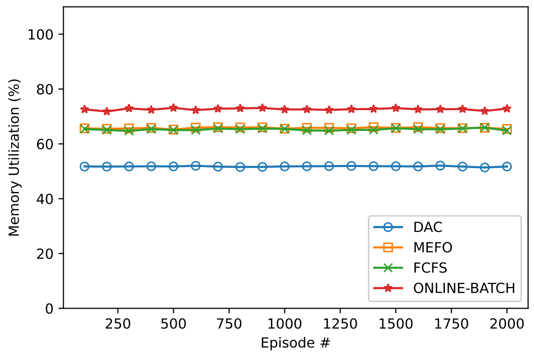
4.3 Experimental Results in Extended Scenarios

4.3.1 Max Occupation Period With 25 Time Slots

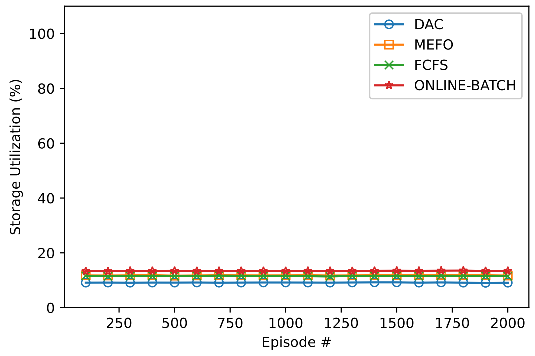
The performance curves of the edge servers using the admission control basing these algorithms processing the offloading requests, whose max occupation period is 25, are plotted in Fig. 4.3. Overall, the DAC's performance in maximizing the system



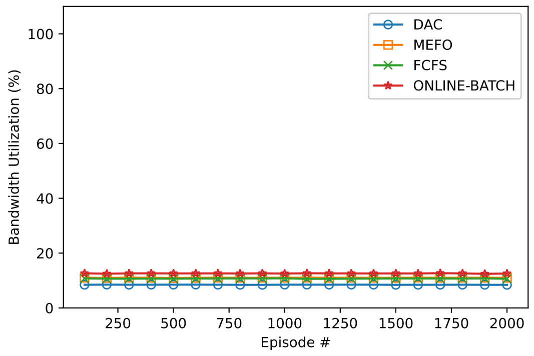
(a) CPU Utilization



(b) Memory Utilization



(c) Disk Utilization



(d) Bandwidth Utilization

Figure 4.2: Resource Utilization with $\tau_{max} = 20$

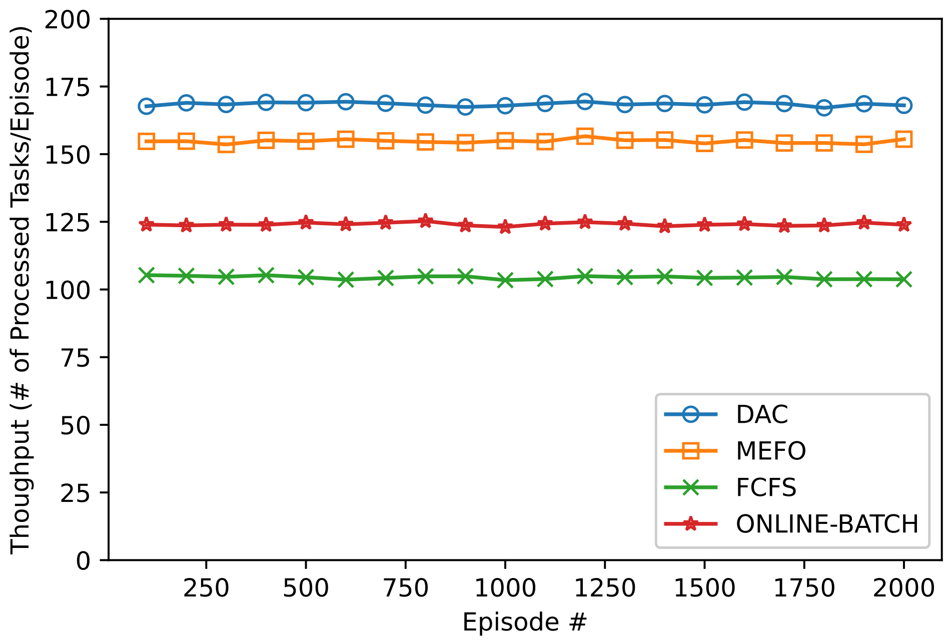


Figure 4.3: System Throughput with $\tau_{max} = 25$

throughput is still better than other admission control algorithms' in this experiment environment. Moreover, comparing with Fig. 4.1, the edge server with the proposed algorithm's throughput decreases by roughly 13, which is the smallest decrease comparing with the other edge servers in the simulation. On the other hand, the edge server applying the MEFO algorithm, whose system throughput is the second-highest in this comparison, loses around 24 units of the system throughput. Though the performance of MEFO is surpassed by neither ONLINE-BATCH nor FCFS, the decrease of system throughput of the edge server implementing MEFO for admission control is the largest. This decrease also indicates the drop of the MEFO's performance in admission control. Though the system throughput of the edge server applying MEFO is 30 larger than the edge server applying ONLINE-BATCH, the MEFO is still possibly performing worse than the ONLINE-BATCH, if its resistance is also worse than ONLINE-BATCH in further simulations.

In the summary the experiments with $\tau_{max} = 25$, according to the specific comparison among four algorithms, the performance of the proposed algorithm is about

61%, 36%, and 9.6% higher than the throughputs of the admission controls with FCFS, ONLINE-BATCH and MEFO respectively. The above digits indicate that the performance loss of the DAC is smaller than its counterparts.

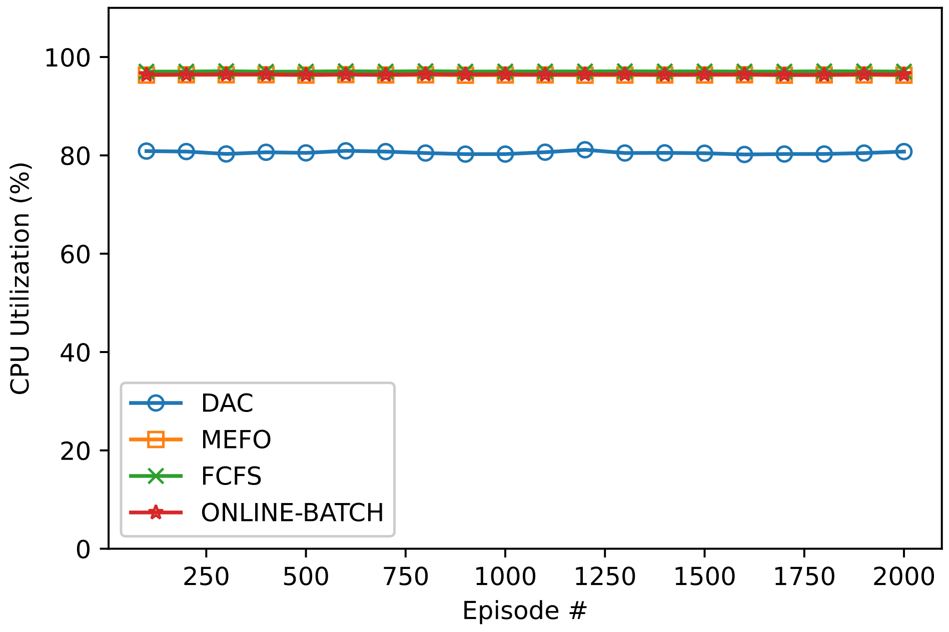


Figure 4.4: CPU Utilization with $\tau_{max} = 25$

When it comes to resource utilization with $\tau_{max} = 25$, the performance of four algorithms can be explained with the Fig. 4.4 and Fig. 4.5. As we observed in basic environment, the CPU is the most significant hardware resource, which strongly influence the throughput performance of edge server. In the current environment, whose max occupation period is 25, the DAC's CPU utilization is roughly stay at 80 %. Meanwhile, the other three algorithm-based admission control's CPU utilizations remain steady at around 99 %. Therefore, while all counterparts are admitting as many requests as possible at every timeslot, the proposed algorithm admits the requests with a careful restraint. Furthermore, the DAC's CPU utilization slightly decrease by 1 %. As for the edge server's memory utilization, a slight decline can be witnessed by comparing the lines of DAC in Fig. 4.2b and Fig. 4.5. In Fig. 4.2b, the edge server with DAC allocates about 50 % of its memory resources on the offloaded requests. In the current environment, the edge server's memory utilization decreases to 48 %. On the contrary, when edge server apply admission control with other three algorithms, the memory utilization increases by 1-2 %.

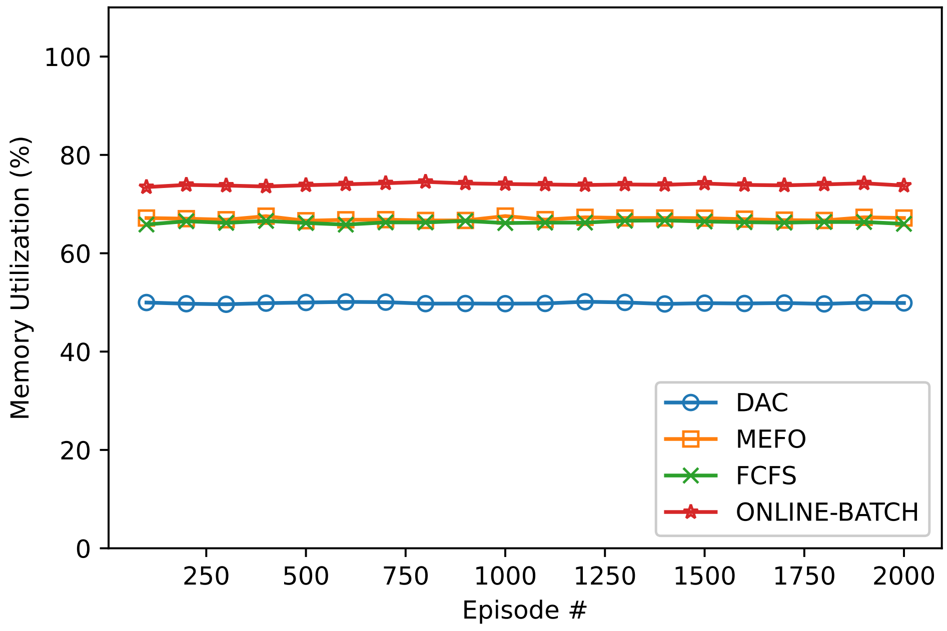


Figure 4.5: Memory Utilization with $\tau_{max} = 25$

As for the edge server's memory utilization, a slight decline can be witnessed by comparing the lines of DAC in Fig. 4.2b and Fig. 4.5. In Fig. 4.2b, the edge server with DAC allocates about 50 % of its memory resources on the offloaded requests. In the current environment, the edge server's memory utilization decreases to 48 %. On the contrary, when edge servers apply admission control with the other three algorithms, the memory utilization increases by 1-2 %.

In the comparison of the simulations in the current environment and the previous environment, we noticed that the overall difference of algorithms' resource allocation is not changed. As for the specific resource utilization in simulations, only slight changes can be spotted. The differences in system utilizations are so small that value fluctuations can be a reasonable explanation for the difference. Therefore, we focus on the changes of specific system utilizations in the incoming discussion about simulations with more different environments.

4.3.2 Max Occupation Period With 30 Time Slots

The performance curves of these algorithms processing the offloading requests, whose max occupation period is 30 are plotted in Fig. 4.6. As expected, the DAC's performance is still the best in the current experiment environment. Moreover, comparing with Fig. 4.3, the edge server with the proposed algorithm's throughput decreases by roughly 14, which is the smallest decrease among all algorithms. On the other hand, the edge server with MEFO, whose system throughput is the second-best among four algorithms, loses around 17 units of the throughput. The decrease of the performance of MEFO in the change of environment configurations is still the largest comparing with other admission control algorithms. On the other hand, comparing with the previous environment, FCFS in the current environment also decreases 14. Despite the MEFO still outperforms the ONLINE-BATCH and the FCFS, the advantage of MEFO to ONLINE-BATCH has become smaller.

To conclude the specific comparison among four algorithms, the performance of the proposed algorithm is about 71%, 41%, and 13% higher than the throughputs of the admission controls basing on FCFS, ONLINE-BATCH and MEFO respectively. The increase of difference between the proposed algorithm and its counterparts suggests, unlike the other admission control solutions, DAC increases its advantage when the incoming requests having longer running time.

When τ_{max} increases to 30, the resource utilization of edge servers applying four algorithms are illustrated in Fig. 4.7 and Fig. 4.8. As we observed before, the CPU is the most significant hardware resource, which strongly influences the throughput performance of the edge server. In the current environment, where the request's max occupation period is 30, the DAC's CPU utilization roughly stays at 79 %. Comparing the previous environment, the CPU utilization of the edge server applying DAC is obviously lower than 80 %. Therefore, an evident decrease in CPU utilization

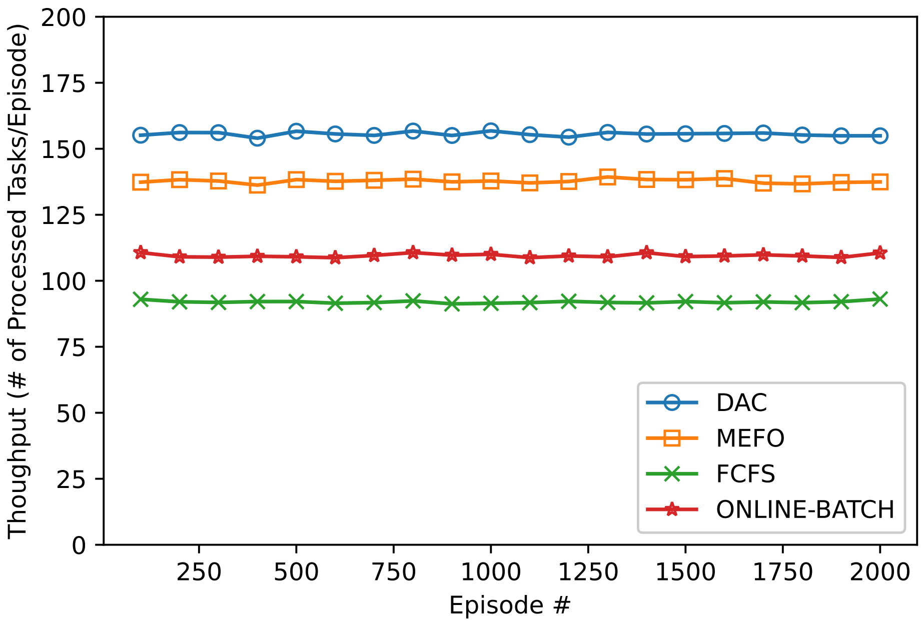


Figure 4.6: System Throughput with $\tau_{max} = 30$

with the DAC algorithm is witnessed. Meanwhile, the CPU utilization of the edge servers applying the other three algorithm-based admission controls remains steady at around 99 %. Therefore, while all counterparts are admitting as many requests as possible at every timeslot, the proposed algorithm admits the requests with more careful restraint, since the incoming requests are more likely to have a longer execution time.

As for the edge server's memory utilization, a slight decline can be witnessed by comparing the lines of DAC in Fig. 4.5 and Fig. 4.8. In Fig. 4.5, the edge server with DAC allocates about 48 % of its memory resources on the offloaded requests. In the current environment, the edge server's memory utilization decreases to 46 %. On the contrary, when the edge server applies admission control with the other three algorithms, the memory utilizations increase by 1 %.

Up to now, the changing trends of resource utilization of the edge servers applying different admission control policies are large enough. Therefore, these changes in system utilization can be claimed as the effect caused by the change of the environment.

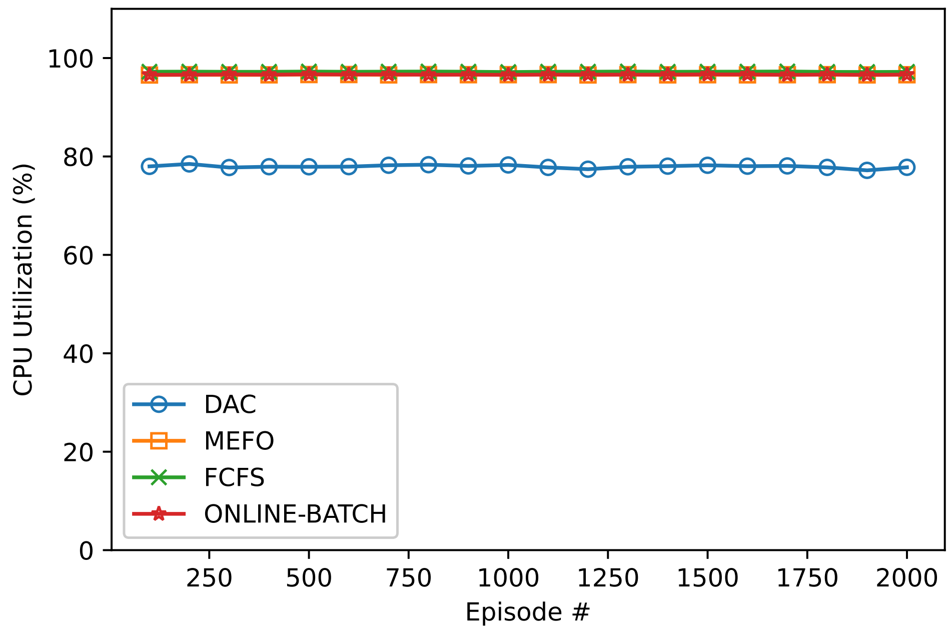


Figure 4.7: CPU Utilization with $\tau_{max} = 30$

Therefore, we can firmly conclude that the edge server applying the DAC is using fewer resources when requests are occupying resources for a longer period of time. On the contrary, the edge servers applying the other three admission control policies increased their system utilization in the same situation.

4.3.3 Max Occupation Period With 35 Time Slots

The performance curves of these algorithms processing the offloading requests, whose max occupation period is 35 are plotted in Fig. 4.9. As expected, the throughput of the edge server applying the DAC is also higher than the edge servers using a different algorithm for admission control. Comparing with Fig. 4.6, the edge server with the proposed algorithm's throughput decreases by roughly 12, which is 3 larger than the FCFS's decrease, which is the smallest among all algorithms. On the other hand, MEFO, whose performance is the second-best among the four algorithms, the edge server applying it loses around 13 units of the throughput. The decrease of

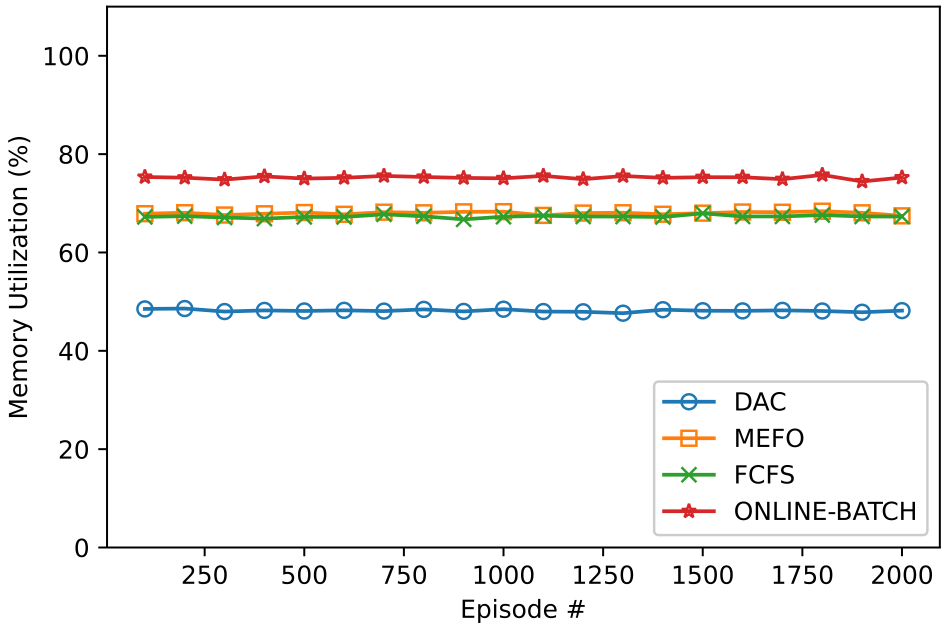


Figure 4.8: Memory Utilization with $\tau_{max} = 30$

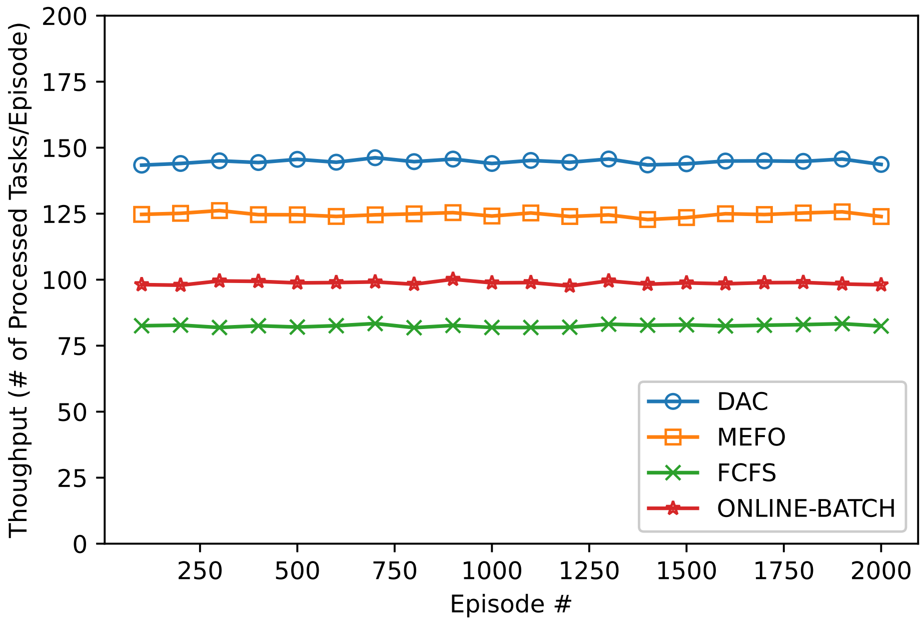


Figure 4.9: System Throughput with $\tau_{max} = 35$

the performance of MEFO is the largest as the decrease in the change of max occupation period from 30 to 35 timeslots. Despite the significant decrease in MEFO's performance, the MEFO still outperforms the ONLINE-BATCH and the FCFS. But MEFO's advantage to them keeps decreasing. Moreover, the throughput decreases of DAC, MEFO and ONLINE-BATCH are close to each other. This may indicate, when $\tau_{max} \leqslant 30$, the resistance of the above three algorithms to a large occupation period is close. Moreover, the differences between the edge servers applying these admission control policies will finally become steady.

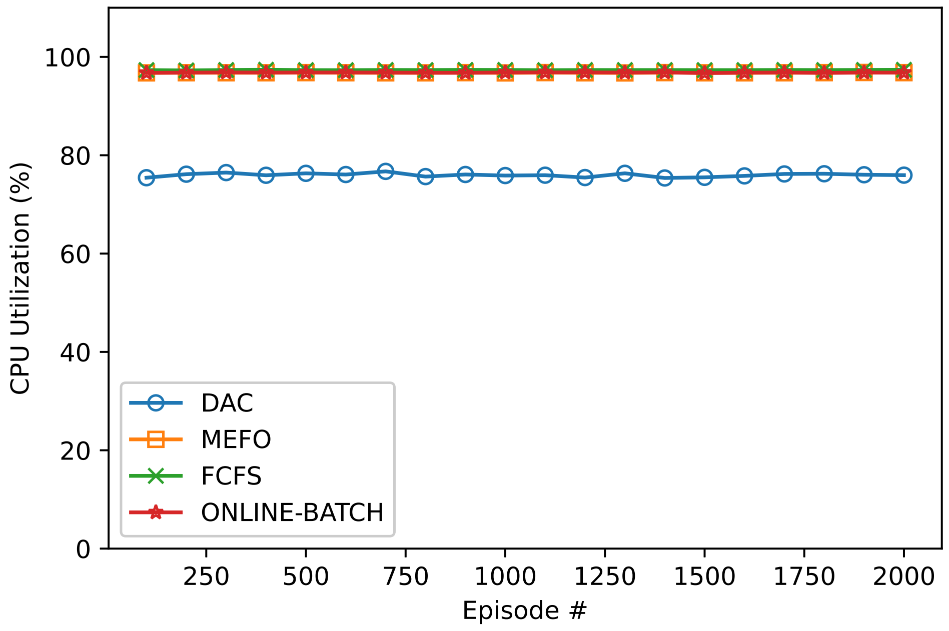


Figure 4.10: CPU Utilization with $\tau_{max} = 35$

To summarize the specific comparison among the four algorithms, the performance of the proposed algorithm is about 75%, 45%, and 15% higher than the throughputs of the admission controls with FCFS, ONLINE-BATCH and MEFO respectively. The increase of difference between the proposed algorithm and its counterparts suggests, DAC still improves its advantage in the current environment. The DAC keeps enlarging its advantages to the other algorithms in the above experiments with the increase of the max running time. Up to now, the DAC mitigates the influence of longer max execution time on the system throughput.

When τ_{max} increases to 35, the resource utilization of edge servers applying four admission control algorithms are illustrated in Fig. 4.10 and Fig. 4.11. As before, the CPU is still the most significant hardware resource, which strongly influences the throughput performance of edge servers. In the current environment, where the request's max occupation period is 35, the DAC's CPU utilization roughly stays at 78 %. Comparing the previous environment, the CPU utilization of the edge server applying DAC is obviously lower than 80%. Therefore, an evident decrease of CPU utilization on the edge server applying the DAC algorithm is witnessed with the increase of the max occupation period. Meanwhile, CPU utilizations of the other three edge servers with other admission control algorithms remain steady at around 99 %. Therefore, while all counterparts are admitting as many requests as possible at every timeslot, the proposed algorithm admits the requests with careful restraint, since the incoming requests are more likely to have a longer running period.

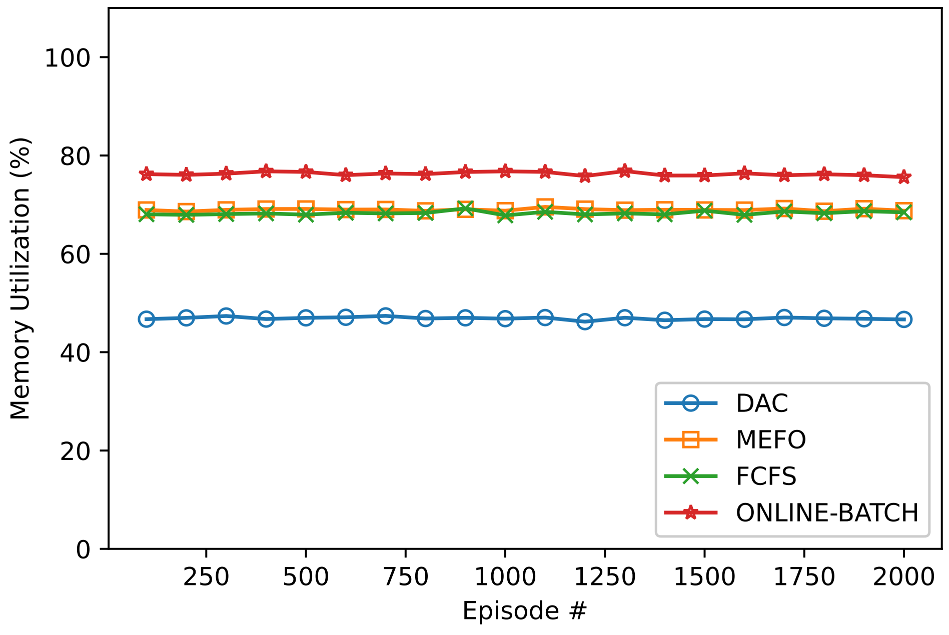


Figure 4.11: Memory Utilization with $\tau_{max} = 35$

As for the edge server's memory utilization, an evident decline can be witnessed since the line of DAC in Fig. 4.11 is much closer to 40 % comparing with the line of DAC in Fig. 4.8. In Fig. 4.8, the edge server implementing DAC allocates about 46 % of its memory resources on the offloaded requests. In the current environment, memory utilization of the edge server, which using DAC to filter incoming requests, decreases to 43 %. On the contrary, the memory utilization of ONLINE-BATCH is much closer to 80% comparing with Fig. 4.8. This comparison firmly proves the increase of memory utilization on the edge server applying admission control with the other three algorithms.

4.3.4 Max Occupation Period With 40 Time Slots

The performance curves of these algorithms processing the offloading requests, whose max occupation period is 40 are plotted in Fig. 4.12. Unexceptionally, the throughput of the edge server applying the DAC is also higher than the edge servers applying a different algorithm for admission control. Comparing with Fig. 4.9, the edge server with the proposed algorithm's throughput decreases by roughly 6, which is the smallest decrease among all algorithms. At the same time, MEFO, the algorithm with the second-best performance, loses around 10 units of the throughput. The decrease of the performance of MEFO is still the largest as the decrease in the change of max occupation period from 35 to 40 timeslots. Though MEFO's performance decreases more than other algorithms, the MEFO still outperforms the ONLINE-BATCH and the FCFS. And since the edge server applying the ONLINE-BATCH only decreases about 9 units, the line of MEFO is not obviously getting close to the line of ONLINE-BATCH. On the other hand, this environment is the first environment whose average throughput decrease is lower than 10. This may indicate, the influence of τ_{max} on the server throughput is not evident as before.

As for the specific comparison among four algorithms, the performance of the proposed algorithm is about 84%, 53%, and 20% higher than the throughputs of the admission controls with FCFS, ONLINE-BATCH and MEFO respectively. To be specific, the system throughput of the edge server applying the DAC is almost a double of the counter part of the edge server only implement the FCFS as admission policy. The increase of difference between the proposed algorithm and its counterparts suggests, DAC still increases its advantage in the current environment.

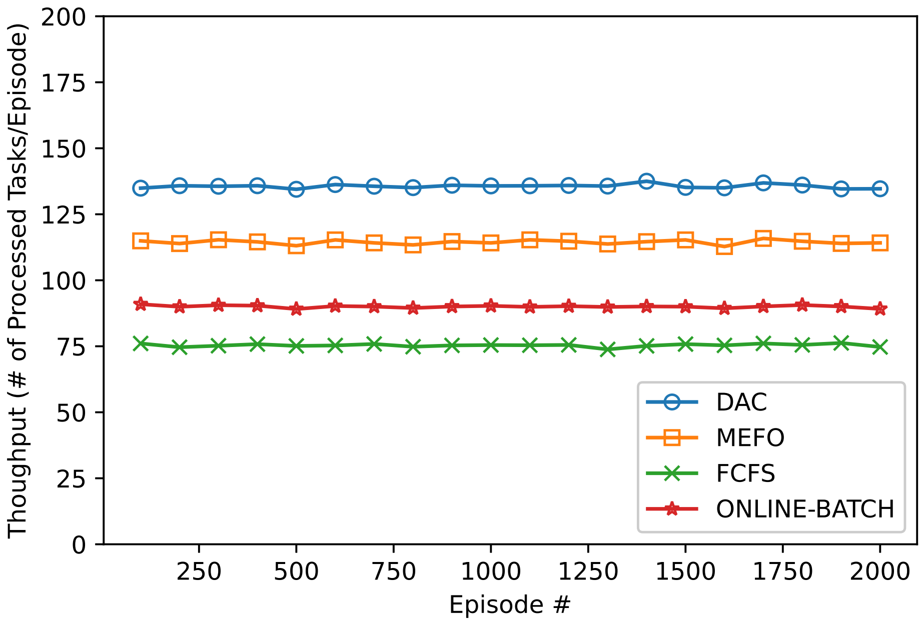


Figure 4.12: System Throughput with $\tau_{max} = 40$

When τ_{max} increases to 40, the resource utilization of edge servers applying four algorithms are illustrated in Fig. 4.13 and Fig. 4.14. As expected, the CPU is as significant as before, because, for most edge servers, the CPU resource is fully utilized in experiments and no more tasks can be admitted. In the current environment, where the request's max occupation period is 40, the DAC's CPU utilization roughly stays at 73 %. Though the difference in CPU utilization between the two environments is estimated to be around 4-6 %, it's hard to compare the lines across the two figures. Nevertheless, comparing the previous environment, the CPU utilization of the edge server applying DAC is obviously much closer to 70%. Therefore, in this comparison of the CPU utilizations in two environments,the decrease of CPU utilization with the lines of DAC algorithm caused by the increase of the max occupation period is witnessed. Meanwhile, the CPU utilizations of the edge servers applying other three algorithm-based admission controls remain steady at around 99 %. This high CPU utilization not only indicates that the edge servers applying the other three admission control algorithms used up the CPU resources but also suggests the cases where incoming tasks can be admitted are few.

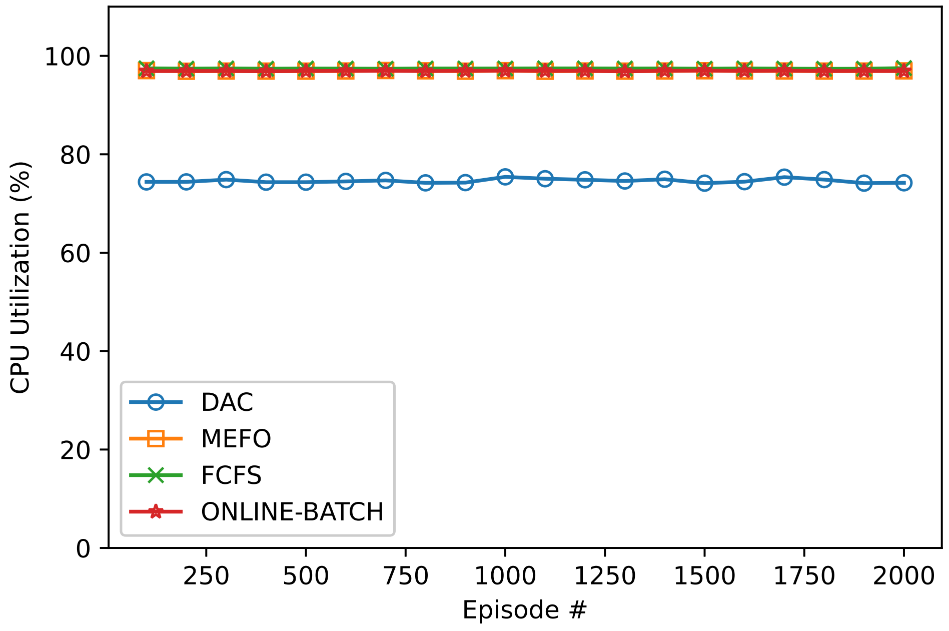


Figure 4.13: CPU Utilization with $\tau_{max} = 40$

As for the edge server's memory utilization, the change in this environment is not as evident as before. In Fig. 4.14, the edge server with DAC also allocates about 43 % of its memory resources on the offloaded requests. On the other hand, the memory utilization of ONLINE-BATCH is much closer to 80 % comparing with Fig. 4.11. This comparison also proves the increase of memory utilization on the edge server applying admission control with the other three algorithms.

4.3.5 Max Occupation Period With 45 Time Slots

The performance curves of these algorithms processing the offloading requests, whose max occupation period is 45 are described in Fig. 4.15. Unexceptionally, the throughput of the edge server applying the DAC is performing better than the edge servers using a different algorithm for admission control. In the comparison between the

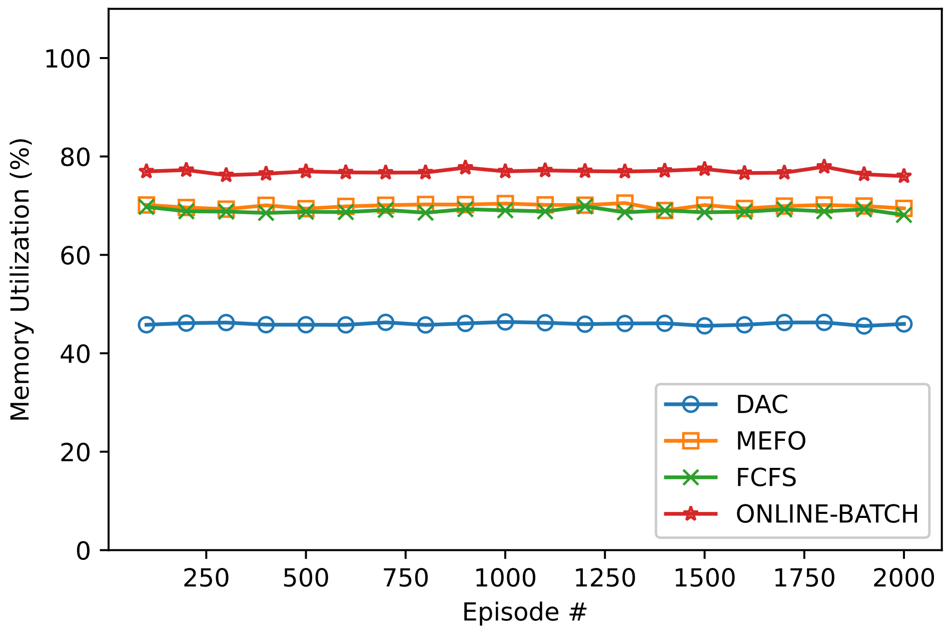


Figure 4.14: Memory Utilization with $\tau_{max} = 40$

current environment and the previous environment, unlike previous comparisons, the edge server applying FCFS decreases the least system throughput, which is about 5. Both edge servers implementing DAC and MEFO for admission control lose 10 units of the system throughput, which is the largest system throughput lost in the change of max occupation period from 40 to 45 timeslots. Even if DAC's and MEFO's performance decreases the most, both of them still outperform the ONLINE-BATCH and the FCFS. On the other hand, the average throughput decrease is also lower than 10. The change of the system throughput caused by the increase of the requests' max occupation time is also obviously mitigated with the growth of the running time of the incoming requests.

As for the specific comparison among four algorithms, the performance of the proposed algorithm is about 82%, 54%, and 21% higher than the throughputs of the admission controls with FCFS, ONLINE-BATCH, and MEFO respectively. The increase of difference between the proposed algorithm and its counterparts suggests, DAC still increases its advantage to ONLINE-BATCH and MEFO in the current

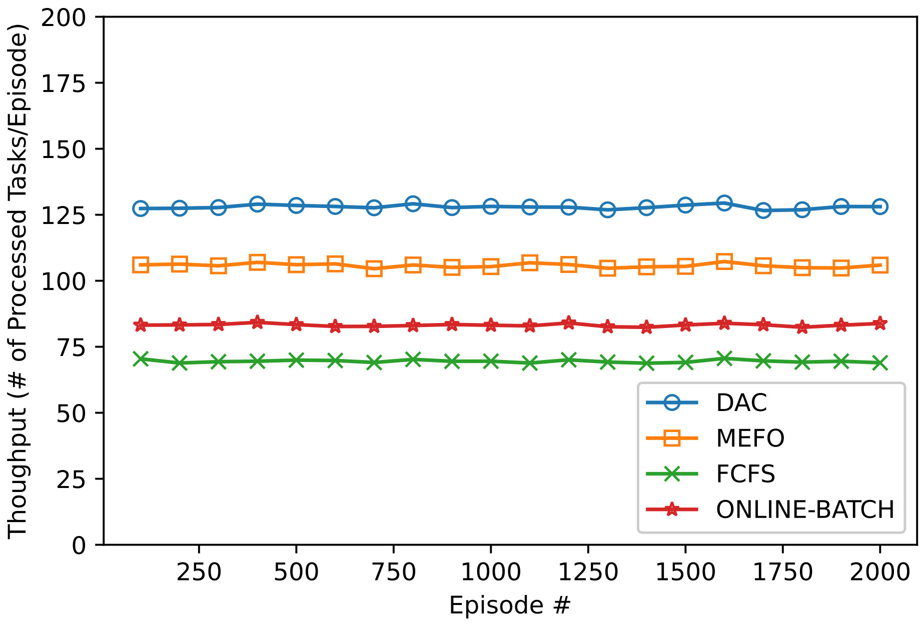


Figure 4.15: System Throughput with $\tau_{max} = 45$

environment. Though the proposed algorithm's advantage to FCFS decreased, this advantage is still the largest among comparisons in the current environment.

When τ_{max} increases to 45, the resource utilization of edge servers applying four algorithms are illustrated with Fig. 4.16 and Fig. 4.17. As expected, the CPU is as significant as before. In the current environment, whose max occupation period is 45, the DAC's CPU utilization roughly stays at 75 %. Comparing the previous environment, the CPU utilization of the edge server applying DAC is much closer to 80 %. Though the decrease of CPU utilization of the edge server with the DAC algorithm is still ambiguous in the comparison between the previous environment and the current environment, comparing with the most previous environment, the decrease of CPU utilization over a long term caused by the increase of the max occupation period is obvious. Meanwhile, the other three algorithm-based admission controls' CPU utilizations remain steady at around 99 %.

As for the edge server's memory utilization, the change in this environment is still not evident. In Fig. 4.17, the edge server with DAC also allocates about 43 %

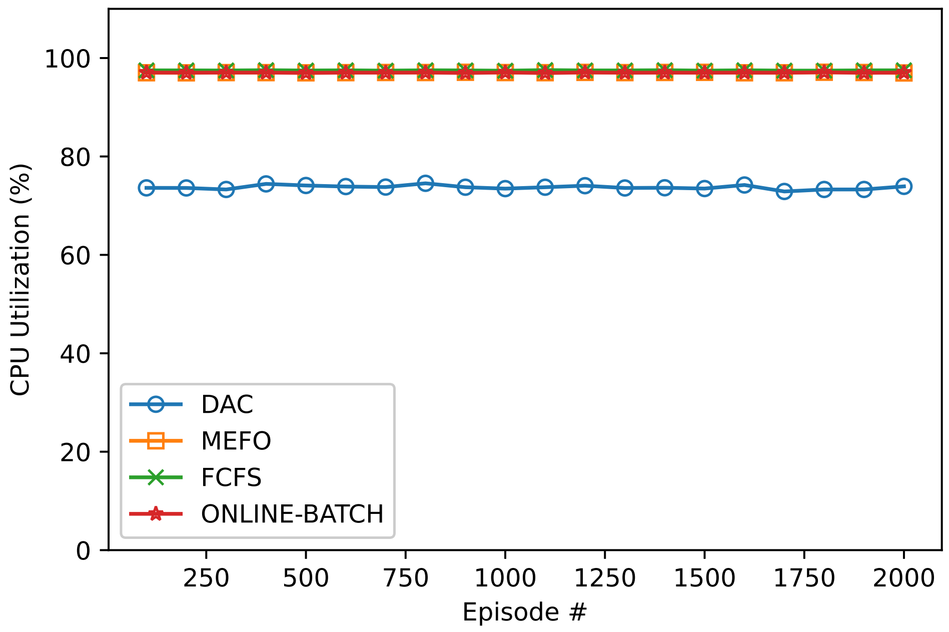


Figure 4.16: CPU Utilization with $\tau_{max} = 45$

of its memory resources on the offloaded requests. On the other hand, the memory utilization of ONLINE-BATCH is much closer to 80% comparing with Fig. 4.14. If comparing with Fig. 4.2b, the increase of the memory utilization is firmly happened on the edge server, which applying the ONLINE-BATCH algorithm for admission control. Up to now, the change direction of the memory utilization of the edge servers applying sorting-based admission control is firmly supported by the comparisons.

4.3.6 Max Occupation Period With 50 Time Slots

The performance curves of these algorithms processing the offloading requests, whose max occupation period is 45 are described in Fig. 4.18. Unexceptionally, the throughput of the edge server applying the DAC is performing better than the edge servers using a different algorithm for admission control. In the comparison between the current environment and the previous environment, like the change in the last environment, the edge server implementing admission control with FCFS decreases the least system throughput, which is about 5. Both edge servers implementing DAC and

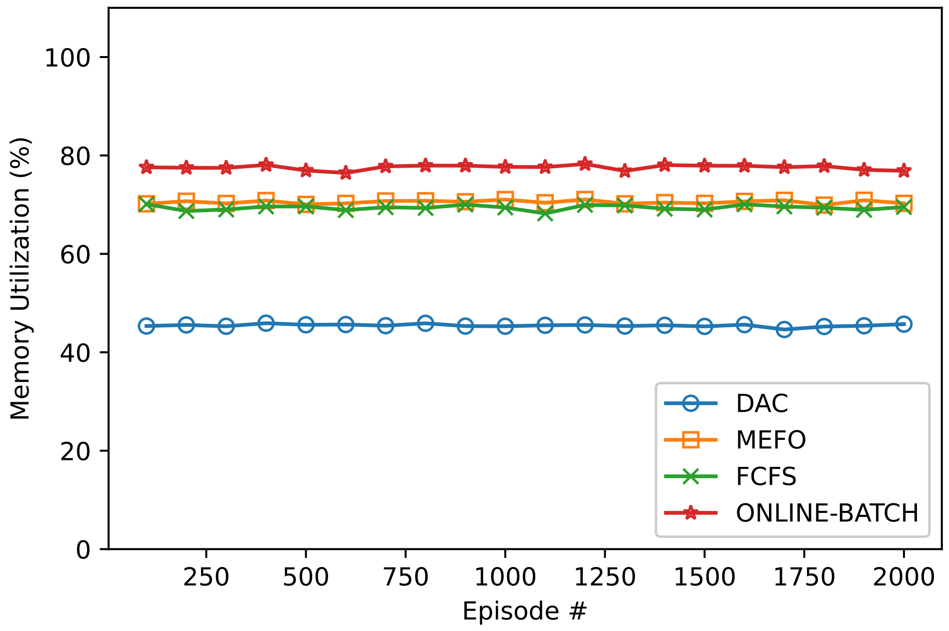


Figure 4.17: Memory Utilization with $\tau_{max} = 45$

MEFO lose 7 units of the system throughput, which is the largest system throughput lost in the change of max occupation period from 45 to 50 timeslots. Even if DAC's and MEFO's performance decreases the most, both of them still outperform the ONLINE-BATCH and the FCFS. On the other hand, the average throughput decrease is also lower than 7. Furthermore, the difference between the edge servers' system throughput lost is smaller than 2. Therefore, none of the algorithms show better resistance to the increase of the max occupation period.

As for the specific comparison among four algorithms, the performance of the proposed algorithm is about to 86%, 57%, and 23% higher than the throughputs of the admission controls with FCFS, ONLINE-BATCH, and MEFO respectively. The increase of difference between the proposed algorithm and its counterparts suggests, DAC also increases its advantage to the admission controls with other algorithms in the current environment. And since, up to now, the max running time has increased from 20 to 50, the DAC is verified that it can keep maintaining and enlarging its advantage to other admission control algorithms.

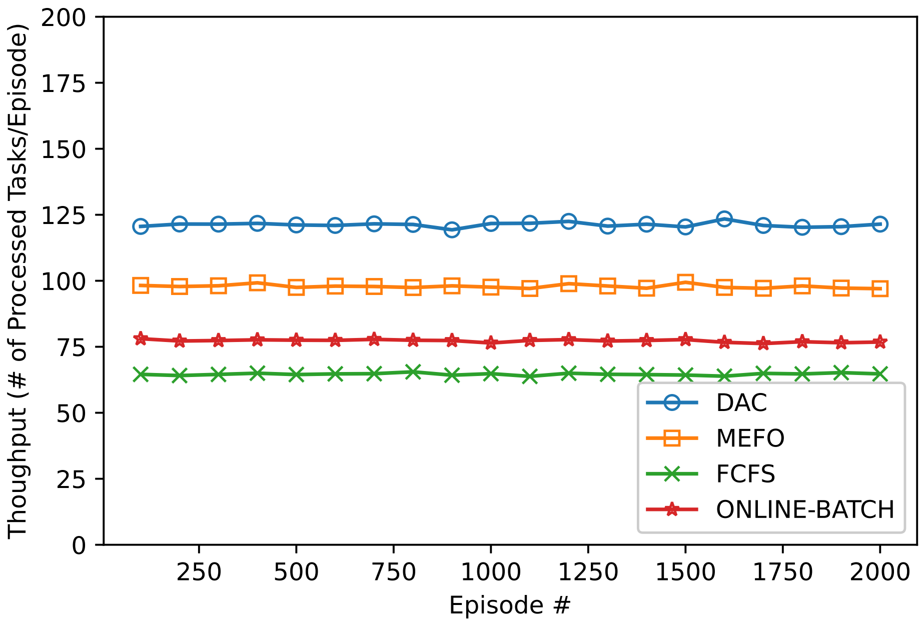


Figure 4.18: System Throughput with $\tau_{max} = 50$

When τ_{max} increases to 50, the resource utilization of edge servers applying four algorithms are illustrated in Fig. 4.19 and Fig. 4.20. As expected, the CPU is as significant as before. In the current environment, whose max occupation period is 50, the DAC's CPU utilization roughly stays at 75 %. Comparing the previous environment, the CPU utilization of the edge server applying DAC is not obviously changed. Meanwhile, the other three algorithm-based admission controls' CPU utilizations remain steady at around 99 %.

As for the edge server's memory utilization, the change in this environment is still not evident. In Fig. 4.20, the edge server with DAC also allocates about 43 % of its memory resources on the offloaded requests. On the other hand, the memory utilization of ONLINE-BATCH is much closer to 80% comparing with Fig. 4.17. Up to now, the change direction of the memory utilization of the edge servers applying sorting-based admission control is fully proven by all changes from $\tau_{max} = 20$ to $\tau_{max} = 50$.

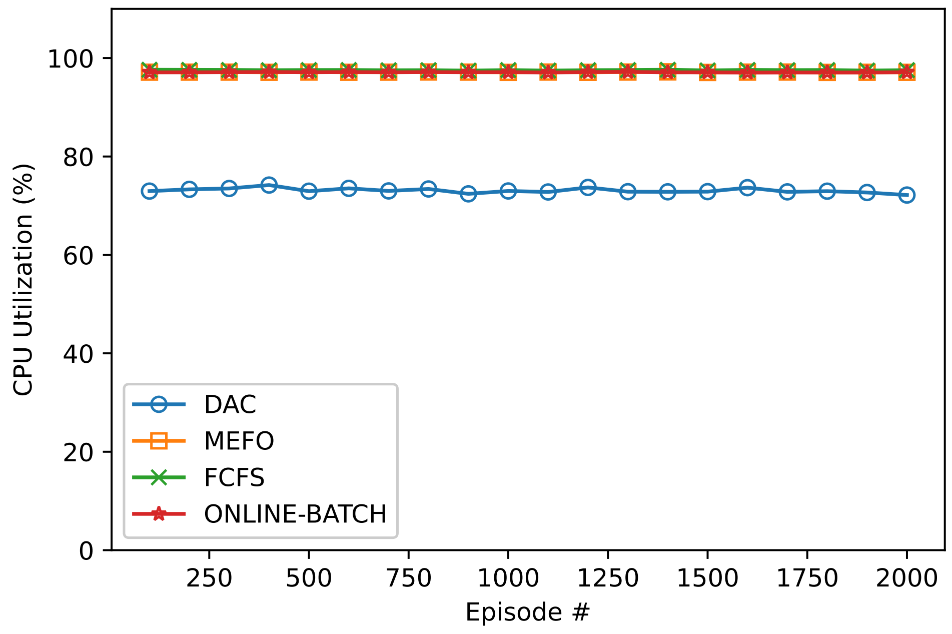


Figure 4.19: CPU Utilization with $\tau_{max} = 50$

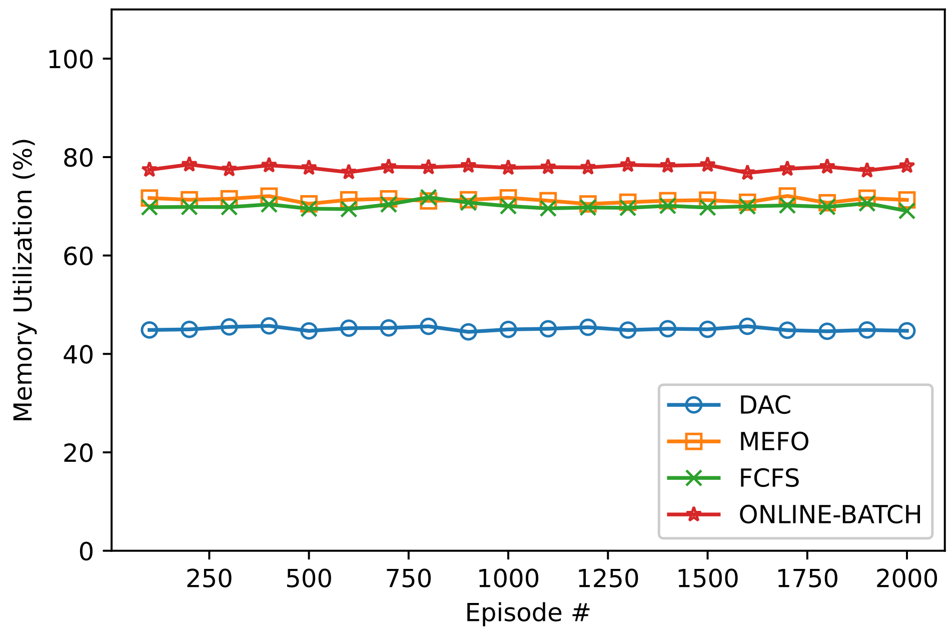


Figure 4.20: Memory Utilization with $\tau_{max} = 50$

4.4 Summary of Throughput Results

The overall performance of the DAC overall experimental environments can be described in Fig. 4.21. According to the experimental data across all predefined environments, the DAC outperforms all other admission control solutions with less consumption of system resources. Moreover, when the max occupation period of request increases, the DAC normally loses the least system throughput. This feature indicates DAC's strong robustness to the requests with a longer execution time.

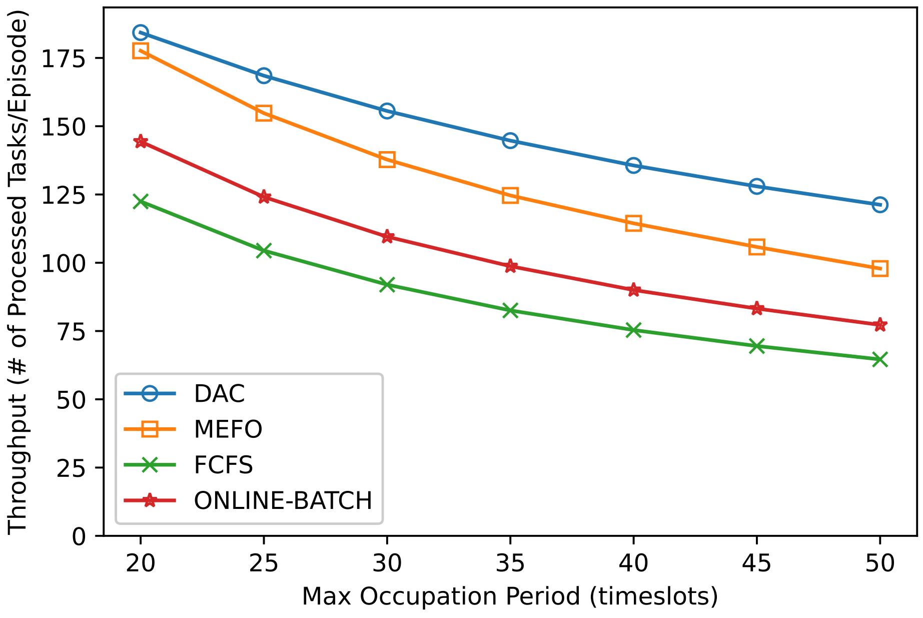


Figure 4.21: Impact of Max Occupation Period

At the same time, we noticed that the influence of max occupation time is not strong anymore when $\tau_{max} > 40$. Furthermore, the differences between the decreases of the algorithms are also dropping accompanying by the increase of the τ_{max}. Therefore, the advantage of the DAC can also be advanced to its counterparts in most inferable environments. On the other hand, if the advantage of the DAC is calculated with the proportion, the superiority of DAC is growing up steadily with the augmentation of the τ_{max}. When $\tau_{max} = 45$, even if the DAC decreases more than the admission control with ONLINE-BATCH and FCFS, the DAC exceeding proportions to those admission control solutions still increase around 1 %. Therefore, the above figures do not only prove the powerful performance comparing with other algorithms, but also illustrates the DAC's solid robustness in handling resource-intensive requests.

In the aspect of the system resource utilization, we noticed the changes of the system utilization that happens on edge servers with DAC are totally different from the other edge servers. Though changes are slight when the difference of τ_{max} is only 5, the decrease of DAC's system resource utilization is obvious if we compare the figures in Section 4.2 with Section 4.3.6. As for the edge servers applying MEFO, ONLINE-BATCH, and FCFS, the trend their system resource utilizations go in the opposite direction of DAC.

In general cases, the fewer spare system resources indicate a better performance of the edge server in accomplishing tasks. However, to receive new incoming requests, the edge server should have sufficient system resources, which meet the requirements of the requests. To be specific, at the time slot where edge sever accepts the requests requiring 20 % of system resource, the edge server's system utilization should not be larger than 80 %. Therefore, the increase of the max occupation period improves the average request running time. This increment of running time influences the frequency of the edge server's request acceptance. Finally, the average system utilization of the edge servers applying sorting-based admission control is increased.

Unlike the sorting-based admission control solutions, the DAC rejects the incoming requests if they are expected to use a large number of resources or occupy resources for a long time. The increase of the max occupation period improves the probability of the edge server handling resource-intensive requests. Therefore, the edge server applying DAC will reject more incoming requests when τ_{max} increases. The more edge server rejects, the fewer system resources are utilized.

Chapter 5

Conclusion and Future Work

This chapter concludes the thesis by analyzing the algorithm mechanism, summarizing the experimental results, comparing the proposed algorithms to other admission control algorithms. Meanwhile, the experimental results also illustrate the space for further improvement. Therefore, the possible space for improvement and the research to be processed in the future are also presented.

5.1 Conclusion

In this paper, we present a novel admission control scheme for MEC, DAC, to maximize the system throughput in MEC. The existing admission control methods are sorting the sequence of the incoming tasks. With these admission control solutions, the edge server admits the requests when it has sufficient resources passively. In this case, the tasks, whose resources requirements are low but execution time is long can still cause negative influence on the edge server's system throughput. Therefore, the DAC's basic mechanism, which is different from the passive admission control solutions, is processing the admission policy actively. Technically, DAC is a DRL-based method, which employs the DDPG algorithm to generate the appropriate admission decisions. By observing the request details of historical tasks and the impact of the tasks on available resources on edge servers, DAC gradually learns the best admission control policy for an edge server in MEC. Furthermore, the tasks, which potentially deduct the edge server's long-term system throughput, can be successfully rejected by the policies generated by the DAC.

In order to evaluate the performance of the DAC, a series of extensive simulation experiments are proposed. These simulations are based on a basic environment, where an edge server with specific resources capacities is established and a set of randomly generated tasks arrive in the edge server at every timeslot. Through these simulations, we compare the performance of DAC to that of the existing admission control methods, namely ONLINE-BATCH, MEFO, and FCFS, in terms of system throughput and resource utilization. Our experimental results indicate DAC leads to the highest system throughput, which is around 183, among the admission control schemes under investigation. Meanwhile, we adjust the max request occupation time for more simulations to see the robustness of these admission control methods. In the further simulations, the DAC not only proves its robustness but also increases its advantage in system throughput to the other admission control algorithms.

As for the resource utilization, the edge server, which applies the DAC for admission control, used fewer hardware resources compared with the edge servers applying other admission control methods. In the further simulations, the resource utilization of the edge server applying the DAC, unlike the other edge servers, decreases with the increase of the task's max occupation time. With further analysis and discussion, we believe that the increase of the task's max occupation time will reduce the possibility of the edge server having free resources to admit new task requests. Therefore, the edge servers applying other admission control solutions increase their system resources utilization. Moreover, since the DAC rejects unsuitable tasks actively, the increase of the max occupation time increases the proportion of the requests rejected by the DAC. At last, the system utilization of the edge server applying the DAC decreases.

To summarize, the DAC outperforms other admission control algorithms in maximizing the edge server's system throughput. Moreover, the DAC's resistance to the increased request execution time is the best, since the DAC's advantage in system throughput is increased with the increase of request execution. Therefore, DAC's performance and robustness in maximizing the throughput are verified by the simulations.

5.2 Future Work

The current solution successfully makes the model learn and evaluate every single request for an admission decision. As a result, this model produces reliable admission policies that outperform the admission policies based on the other admission control algorithms. Though the proposed solution increases the edge server performance in maximizing throughput, about 20 % CPU is not utilized. This spare space of system utilization potentially increases the edge server's system throughput, by leaving sufficient hardware resources for more resource-friendly requests in the next time slots. Meanwhile, this unused proportion of the CPU resource also indicates room for further maximizing the system throughput. Therefore, if the current model can involve the consideration of utilizing as much hardware as possible, the current solution's benefits to the edge server can be further improved.

On the other hand, the performance of trained models is not so stable that it can not always guide the MEC to reach the largest system throughput. Considering the mechanism of the neural network, we believe that the current input of the neural network is not enough for the model to observe the status of the system. Therefore, we expect to involve more observable variables, such as the queue of the tasks running on the edge server. Furthermore, by observing the failed admission policies, we realized that though we constructed a reward system for the admitted request, this reward system may mislead the neural network because of the contribution of the request characteristics. If the neural network is misled by this reward, the major function of the network may be trained into a filter, which makes admission decisions only according to the request characteristics. Therefore, we can further develop the reward calculation logic to ensure the long-term system throughput is the major contributor to the final reward to improve the performance of the DAC.

Bibliography

[1] A. Al-Shuwaili and O. Simeone, “Energy-efficient resource allocation for mobile edge computing-based augmented reality applications,” *IEEE Wireless Communications Letters*, vol. 6, no. 3, pp. 398–401, 2017.

[2] L. Liu, Y. Zhou, J. Yuan, W. Zhuang, and Y. Wang, “Economically optimal ms association for multimedia content delivery in cache-enabled heterogeneous cloud radio access networks,” *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 7, pp. 1584–1593, 2019.

[3] Y. Zhou, L. Tian, L. Liu, and Y. Qi, “Fog computing enabled future mobile communication networks: A convergence of communication and computing,” *IEEE Communications Magazine*, vol. 57, no. 5, pp. 20–27, 2019.

[4] Q. Zhang, M. Lin, L. T. Yang, Z. Chen, S. U. Khan, and P. Li, “A double deep q-learning model for energy-efficient edge scheduling,” *IEEE Transactions on Services Computing*, vol. 12, no. 5, pp. 739–749, 2018.

[5] U. Varshney, “4g wireless networks,” *IT Professional*, vol. 14, no. 5, pp. 34–39, 2012.

[6] P. Mogensen, K. Pajukoski, E. Tiirola, E. Lähetkangas, J. Vihriälä, S. Vesterinen, M. Laitila, G. Berardinelli, G. W. Da Costa, L. G. Garcia *et al.*, “5g small cell optimized radio design,” in *2013 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2013, pp. 111–116.

[7] N. Jain and S. Choudhary, “Overview of virtualization in cloud computing,” in *2016 Symposium on Colossal Data Analysis and Networking (CDAN)*. IEEE, 2016, pp. 1–4.

[8] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” *arXiv preprint arXiv:1509.02971*, 2015.

[9] M. Satyanarayanan, “Mobile computing: the next decade,” in *Proceedings of the 1st ACM workshop on mobile cloud computing & services: social networks and beyond*, 2010, pp. 1–6.

[10] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the internet of things,” in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, 2012, pp. 13–16.

[11] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[12] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.

[13] L. Wang, G. Von Laszewski, A. Younge, X. He, M. Kunze, J. Tao, and C. Fu, "Cloud computing: a perspective study," *New generation computing*, vol. 28, no. 2, pp. 137–146, 2010.

[14] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in *2011 IEEE 4th International Conference on Cloud Computing*. IEEE, 2011, pp. 500–507.

[15] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[16] S. Guo, J. Liu, Y. Yang, B. Xiao, and Z. Li, "Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing," *IEEE Transactions on Mobile Computing*, vol. 18, no. 2, pp. 319–333, 2018.

[17] Y. Geng, Y. Yang, and G. Cao, "Energy-efficient computation offloading for multicore-based mobile devices," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 46–54.

[18] J. Xu, L. Chen, and S. Ren, "Online learning for offloading and autoscaling in energy harvesting mobile edge computing," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 3, pp. 361–373, 2017.

[19] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4005–4018, 2018.

[20] H. Huang, Q. Ye, and H. Du, "Reinforcement learning based offloading for real-time applications in mobile edge computing," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[21] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3133–3174, 2019.

[22] J. Li, H. Gao, T. Lv, and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for mec," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2018, pp. 1–6.

[23] Y. Liu, H. Yu, S. Xie, and Y. Zhang, “Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks,” *IEEE Transactions on Vehicular Technology*, vol. 68, no. 11, pp. 11 158–11 168, 2019.

[24] X. Deng, J. Li, L. Shi, Z. Wei, X. Zhou, and J. Yuan, “Wireless powered mobile edge computing: Dynamic resource allocation and throughput maximization,” *IEEE Transactions on Mobile Computing*, 2020.

[25] S. Ulukus, A. Yener, E. Erkip, O. Simeone, M. Zorzi, P. Grover, and K. Huang, “Energy harvesting wireless communications: A review of recent advances,” *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 3, pp. 360–381, 2015.

[26] J. Tang, J. Song, J. Ou, J. Luo, X. Zhang, and K.-K. Wong, “Minimum throughput maximization for multi-uav enabled wpcn: A deep reinforcement learning method,” *IEEE Access*, vol. 8, pp. 9124–9132, 2020.

[27] C. Liang and F. R. Yu, “Wireless network virtualization: A survey, some research issues and challenges,” *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 358–380, 2014.

[28] S. Van Rossem, W. Tavernier, B. Sonkoly, D. Colle, J. Czentye, M. Pickavet, and P. Demeester, “Deploying elastic routing capability in an sdn/nfv-enabled environment,” in *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*. IEEE, 2015, pp. 22–24.

[29] Y. Ma, W. Liang, J. Wu, and Z. Xu, “Throughput maximization of nfv-enabled multicasting in mobile edge cloud networks,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 2, pp. 393–407, 2020.

[30] Y. Yue, B. Cheng, B. Li, M. Wang, and X. Liu, “Throughput optimization vnf placement for mapping sfc requests in mec-nfv enabled networks,” in *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom ’20. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3372224.3418169

[31] Q. Xia, W. Liang, and W. Xu, “Throughput maximization for online request admissions in mobile cloudlets,” in *38th Annual IEEE Conference on Local Computer Networks*. IEEE, 2013, pp. 589–596.

[32] M. Hu, L. Zhuang, D. Wu, Y. Zhou, X. Chen, and L. Xiao, “Learning driven computation offloading for asymmetrically informed edge computing,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 8, pp. 1802–1815, 2019.

[33] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT press, 2018.

[34] C. J. Watkins and P. Dayan, “Q-learning,” *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[35] H. Hasselt, “Double q-learning,” *Advances in neural information processing systems*, vol. 23, pp. 2613–2621, 2010.

[36] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[37] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, “Human-level control through deep reinforcement learning,” *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[38] V. R. Konda and J. N. Tsitsiklis, “Actor-critic algorithms,” in *Advances in neural information processing systems*, 2000, pp. 1008–1014.

[39] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, “Asynchronous methods for deep reinforcement learning,” in *International conference on machine learning.* PMLR, 2016, pp. 1928–1937.

[40] Y. Liu, C. Yang, L. Jiang, S. Xie, and Y. Zhang, “Intelligent edge computing for iot-based energy management in smart cities,” *IEEE network*, vol. 33, no. 2, pp. 111–117, 2019.

[41] B. Chen, J. Wan, Y. Lan, M. Imran, D. Li, and N. Guizani, “Improving cognitive ability of edge intelligent iiot through machine learning,” *IEEE Network*, vol. 33, no. 5, pp. 61–67, 2019.

[42] A. Kanawaday and A. Sane, “Machine learning for predictive maintenance of industrial machines using iot sensor data,” in *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS).* IEEE, 2017, pp. 87–90.

[43] X. Wang, Y. Han, V. C. Leung, D. Niyato, X. Yan, and X. Chen, “Convergence of edge computing and deep learning: A comprehensive survey,” *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 869–904, 2020.

[44] Z. Ning, K. Zhang, X. Wang, L. Guo, X. Hu, J. Huang, B. Hu, and R. Y. Kwok, “Intelligent edge computing in internet of vehicles: a joint computation offloading and caching solution,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 4, pp. 2212–2225, 2020.

[45] B. Fekade, T. Maksymyuk, M. Kyryk, and M. Jo, “Probabilistic recovery of incomplete sensed data in iot,” *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2282–2292, 2017.

[46] T. Wang, J. Zhou, A. Liu, M. Z. A. Bhuiyan, G. Wang, and W. Jia, "Fog-based computing and storage offloading for data synchronization in iot," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4272–4282, 2018.

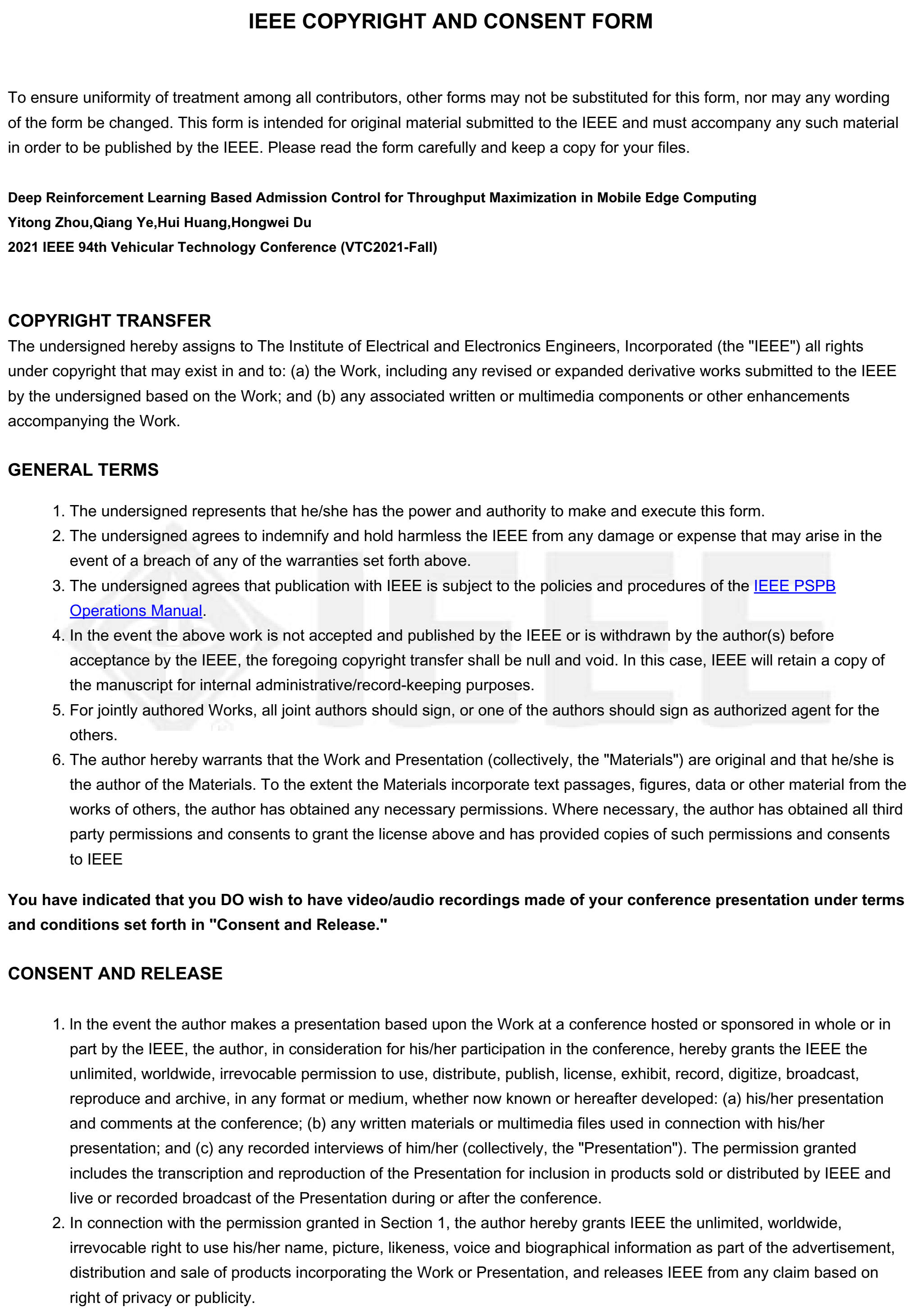
[47] M. Chen, Y. Hao, K. Lin, Z. Yuan, and L. Hu, "Label-less learning for traffic control in an edge network," *IEEE Network*, vol. 32, no. 6, pp. 8–14, 2018.

[48] A. Zappone, M. Di Renzo, and M. Debbah, "Wireless networks design in the era of deep learning: Model-based, ai-based, or both?" *IEEE Transactions on Communications*, vol. 67, no. 10, pp. 7331–7376, 2019.

[49] D. Muirhead, M. A. Imran, and K. Arshad, "Insights and approaches for low-complexity 5g small-cell base-station design for indoor dense networks," *IEEE access*, vol. 3, pp. 1562–1572, 2015.

Appendix A

Copyright Permissions



IEEE COPYRIGHT AND CONSENT FORM

To ensure uniformity of treatment among all contributors, other forms may not be substituted for this form, nor may any wording of the form be changed. This form is intended for original material submitted to the IEEE and must accompany any such material in order to be published by the IEEE. Please read the form carefully and keep a copy for your files.

Deep Reinforcement Learning Based Admission Control for Throughput Maximization in Mobile Edge Computing
Yitong Zhou,Qiang Ye,Hui Huang,Hongwei Du
2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall)

COPYRIGHT TRANSFER

The undersigned hereby assigns to The Institute of Electrical and Electronics Engineers, Incorporated (the "IEEE") all rights under copyright that may exist in and to: (a) the Work, including any revised or expanded derivative works submitted to the IEEE by the undersigned based on the Work; and (b) any associated written or multimedia components or other enhancements accompanying the Work.

GENERAL TERMS

1. The undersigned represents that he/she has the power and authority to make and execute this form.
2. The undersigned agrees to indemnify and hold harmless the IEEE from any damage or expense that may arise in the event of a breach of any of the warranties set forth above.
3. The undersigned agrees that publication with IEEE is subject to the policies and procedures of the IEEE PSPB Operations Manual.
4. In the event the above work is not accepted and published by the IEEE or is withdrawn by the author(s) before acceptance by the IEEE, the foregoing copyright transfer shall be null and void. In this case, IEEE will retain a copy of the manuscript for internal administrative/record-keeping purposes.
5. For jointly authored Works, all joint authors should sign, or one of the authors should sign as authorized agent for the others.
6. The author hereby warrants that the Work and Presentation (collectively, the "Materials") are original and that he/she is the author of the Materials. To the extent the Materials incorporate text passages, figures, data or other material from the works of others, the author has obtained any necessary permissions. Where necessary, the author has obtained all third party permissions and consents to grant the license above and has provided copies of such permissions and consents to IEEE

You have indicated that you DO wish to have video/audio recordings made of your conference presentation under terms and conditions set forth in "Consent and Release."

CONSENT AND RELEASE

1. In the event the author makes a presentation based upon the Work at a conference hosted or sponsored in whole or in part by the IEEE, the author, in consideration for his/her participation in the conference, hereby grants the IEEE the unlimited, worldwide, irrevocable permission to use, distribute, publish, license, exhibit, record, digitize, broadcast, reproduce and archive, in any format or medium, whether now known or hereafter developed: (a) his/her presentation and comments at the conference; (b) any written materials or multimedia files used in connection with his/her presentation; and (c) any recorded interviews of him/her (collectively, the "Presentation"). The permission granted includes the transcription and reproduction of the Presentation for inclusion in products sold or distributed by IEEE and live or recorded broadcast of the Presentation during or after the conference.
2. In connection with the permission granted in Section 1, the author hereby grants IEEE the unlimited, worldwide, irrevocable right to use his/her name, picture, likeness, voice and biographical information as part of the advertisement, distribution and sale of products incorporating the Work or Presentation, and releases IEEE from any claim based on right of privacy or publicity.

BY TYPING IN YOUR FULL NAME BELOW AND CLICKING THE SUBMIT BUTTON, YOU CERTIFY THAT SUCH ACTION CONSTITUTES YOUR ELECTRONIC SIGNATURE TO THIS FORM IN ACCORDANCE WITH UNITED STATES LAW, WHICH AUTHORIZES ELECTRONIC SIGNATURE BY AUTHENTICATED REQUEST FROM A USER OVER THE INTERNET AS A VALID SUBSTITUTE FOR A WRITTEN SIGNATURE.

Qiang Ye	30-07-2021
Signature	**Date (dd-mm-yyyy)**

Information for Authors

AUTHOR RESPONSIBILITIES

The IEEE distributes its technical publications throughout the world and wants to ensure that the material submitted to its publications is properly available to the readership of those publications. Authors must ensure that their Work meets the requirements as stated in section 8.2.1 of the IEEE PSPB Operations Manual, including provisions covering originality, authorship, author responsibilities and author misconduct. More information on IEEE’s publishing policies may be found at http://www.ieee.org/publications_standards/publications/rights/authorrightsresponsibilities.html Authors are advised especially of IEEE PSPB Operations Manual section 8.2.1.B12: "It is the responsibility of the authors, not the IEEE, to determine whether disclosure of their material requires the prior consent of other parties and, if so, to obtain it." Authors are also advised of IEEE PSPB Operations Manual section 8.1.1B: "Statements and opinions given in work published by the IEEE are the expression of the authors."

RETAINED RIGHTS/TERMS AND CONDITIONS

- Authors/employers retain all proprietary rights in any process, procedure, or article of manufacture described in the Work.
- Authors/employers may reproduce or authorize others to reproduce the Work, material extracted verbatim from the Work, or derivative works for the author's personal use or for company use, provided that the source and the IEEE copyright notice are indicated, the copies are not used in any way that implies IEEE endorsement of a product or service of any employer, and the copies themselves are not offered for sale.
- Although authors are permitted to re-use all or portions of the Work in other works, this does not include granting third-party requests for reprinting, republishing, or other types of re-use.The IEEE Intellectual Property Rights office must handle all such third-party requests.
- Authors whose work was performed under a grant from a government funding agency are free to fulfill any deposit mandates from that funding agency.

AUTHOR ONLINE USE

- **Personal Servers**. Authors and/or their employers shall have the right to post the accepted version of IEEE-copyrighted articles on their own personal servers or the servers of their institutions or employers without permission from IEEE, provided that the posted version includes a prominently displayed IEEE copyright notice and, when published, a full citation to the original IEEE publication, including a link to the article abstract in IEEE Xplore. Authors shall not post the final, published versions of their papers.
- **Classroom or Internal Training Use.** An author is expressly permitted to post any portion of the accepted version of his/her own IEEE-copyrighted articles on the author's personal web site or the servers of the author's institution or company in connection with the author's teaching, training, or work responsibilities, provided that the appropriate copyright, credit, and reuse notices appear prominently with the posted material. Examples of permitted uses are lecture materials, course packs, e-reserves, conference presentations, or in-house training courses.
- **Electronic Preprints.** Before submitting an article to an IEEE publication, authors frequently post their manuscripts to their own web site, their employer's site, or to another server that invites constructive comment from colleagues. Upon submission of an article to IEEE, an author is required to transfer copyright in the article to IEEE, and the author must update any previously posted version of the article with a prominently displayed IEEE copyright notice. Upon publication of an article by the IEEE, the author must replace any previously posted electronic versions of the article with either (1) the full citation to the