Edge-Cloud Offloading: Knapsack Potential Game in 5G Multi-Access Edge Computing

Cheng-Ying Hsieh, Yi Ren, Jyh-Cheng Chen, Fellow, IEEE

Abstract—In 5G, multi-access edge computing enables the applications to be offloaded to near-end edge servers for faster response. According to the 3GPP standards, users in 5G are separated into many types, e.g., vehicles, AR/VR, IoT devices, etc. Specifically, the high-priority traffic can preempt edge resources to guarantee the service quality. However, even if a traffic is transmitted with low priority, its latency requirement in 5G is much lower than that in 4G. Too strict latency requirement and priority-based service make resource configuration difficult on the edge side. Therefore, we propose the edge-cloud offloading mechanism, in which each edge server can offload tasks to back-end cloud server to ensure service quality of both high- and low-priority traffic. In this paper, we establish a priority-based queuing system to model the edge-cloud offloading behaviors. Based on the formulation of our system model, we propose Knapsack Potential Game (KPG) to derive an optimal offloading ratio for each edge server to balance the cost-effectiveness of the overall system. We demonstrate that KPG has low computational complexity and outperforms two baseline algorithms. The results indicate that KPG's performance is optimal and provides a theoretical guideline to operators while designing their edge-cloud offloading strategies without large-scale implementation.

Index Terms—Multi-Access Edge Computing, QoS, 5G, Performance Analysis, 3GPP Standards

I. Introduction

WITH the development of the 5th generation mobile networks (5G), the cost-effectiveness of deploying infrastructures for multi-access edge computing (MEC) is now challenging operators. According to 3rd Generation Partnership Project (3GPP) 23.203 [1], the priority of quality of service (QoS) falls into many categories which include ultra-reliable low latency (URLLC) and enhanced mobile broadband (eMBB). URLLC is considered as a high-priority QoS job [2] which is used for smart grids, auto-driving, remote surgery, etc. On the contrary, eMBB is regarded as a low-priority QoS job which is used for Augmented Reality (AR)/ Virtual Reality (VR) media, Ultra High Definition (UltraHD), 360-degree streaming video, etc. According to International Telecommunication Union-Radiocommunication Sector (ITU-R) [3], the requirements on user plane latency are 4 ms for eMBB and 1 ms for URLLC. With the increase in the requirement of low-latency applications, MEC operators need to deploy more edge servers and put them close to users. However, the benefits brought by current edge servers are limited due to 5G priority-based transmission mechanism.

In 5G, 3GPP introduces more features on QoS jobs which makes the resource configuration on the edge side becomes more challenge than that in 4G. That is, the service quality of high-priority QoS jobs is guaranteed but that of low-priority ones is limited. Specifically, QoS jobs are labeled with a 5G QoS Identifier (5QI), making QoS jobs scheduled with their priorities during the transmission. According to 5G new radio (5GNR) [4], [5], regardless of how many types of QoS jobs are transmitted at the same time, QoS jobs are sent to the near-end edge server through a protocol data unit (PDU) session. Since the scheduling mechanism conducted on edge servers has not been specified in 3GPP standards, the challenge is how to schedule the offloading traffic and to also take different types of QoS jobs into consideration. The existing scheduling studies fall into two-fold: (1) Priority scheduling mechanism [6]–[8], and (2) Resource allocation optimization [9]–[11].

1) Priority scheduling mechanism: The authors in [6] proposed a priority-based job scheduling policy to optimize resource allocation in terms of computation and communication costs. This scheme, however, runs in a non-preemptive manner, which is difficult to guarantee the performance of high-priority QoS jobs. The authors of [7] propose a queuing model which separates offloaded jobs into different priority queues. Although multiple queues can execute jobs in the order of priorities, high-priority jobs still need to wait for the completion of low-priority ones which are being served. On the other hand, the authors in [8] proposed priori offloading mechanism with joint offloading proportion and transmission (PROMOT) to deal with the scenario that user devices are covered by multiple edge servers; however, the scheduling policy conducted in this method only focuses on choosing an edge server, ignoring the priority of jobs.
2) Resource allocation optimization: The authors of [9] combine fog servers as the back-end computing resource, optimizing resource allocation with game theory perspectives. However, the jobs discussed in [9] are the same priority that deviates from the

Manuscript received January 7, 2022; revised June 30, 2022, and December 2, 2022; accepted February 8, 2023.

C.-Y. Hsieh is with the Department of Computer Science, National Yang Ming Chiao Tung University (NYCU), Hsinchu, Taiwan. E-mail: ingyaya36.cs03@nycu.edu.tw.

Y. Ren is with the School of Computing Science, University of East Anglia (UEA), Norwich, U.K. E-mail: e.ren@uea.ac.uk.

J.-C. Chen is with the Department of Computer Science, National Yang Ming Chiao Tung University (NYCU), Hsinchu, Taiwan. E-mail: jcc@nycu.edu.tw.

requirements of the 5G MEC system. In addition, the authors of [10] propose a multi-access MEC servers system that optimizes resource allocation on both communication and computation. However, the metric considered in this paper is only energy consumption, while performance optimization is not discussed. In [11], the authors provide services for both edge servers and remote clouds; however, the authors mainly focus on communication cost, neglecting execution costs and idleness costs.

According to the gap between the current studies and the requirements of MEC system, we propose the edge-cloud offloading mechanism, which not only guarantees the QoS for high-/low-priority QoS jobs but also optimizes the cost-effectiveness of the overall system.

Specifically, we integrate the PREEMPT_RT [12] into the offloading mechanism on edge servers, where high-priority QoS jobs can preempt the computing resource on edge servers, while low-priority QoS jobs can be fractionally offloaded to a shared cloud server (SCS). By this way, the new design not only guarantees the QoS for high-/low-priority QoS jobs but also controls the resource allocation for the system. During the design of the edge-cloud offloading system, we encountered several challenges that made this work difficult but worthy of study. In the following statements, we list our major contributions and introduce them separately:

- Avoid the resource competition on SCS:
  The problem of resource allocation on the SCS is different from that in traditional cloud computing. Generally, operators will be notified in advance when there are vast numbers of service requests (e.g., concert tickets, homecoming tickets, anniversary promotions, etc.). In 5G, however, the traffic is more unpredictable. The resource of the SCS should be appropriately allocated for edge servers based on their current situation. Therefore, we propose Knapsack Potential Game (KPG) to address the resource competition problem.
- Estimate the impact of preemption on each edge:
  As discussed above, preemption is a key characteristic in edge-cloud offloading. In this paper, we design a priority queuing model and take the offloading issue into consideration. With the priority queuing model, operators can analyze their offloading strategies in theoretical guidelines without large-scale implementation, saving both cost and time.
- Evaluate the cost-effectiveness of the system:
  To balance the trade-off between cost and performance, we introduce the cost response-time index (CRI) as the objective function. By adjusting the performance bias of the CRI, operators can customize their objective function according to their strategies.
- Determine the number of SCS instances:
  The proposed KPG algorithm can calculate the optimal offloading ratios of edges under different numbers of cloud instances. Moreover, because of the CRI, the original trade-off problem between the cost and response time can be transformed into a convex function, in which we can find the most cost-effective number of cloud instances.

With the above features, operators can find the optimal offloading ratio for each edge to avoid the impact of both preemption and competition. Moreover, the KPG algorithm can further find the optimal number of SCS instances to balance the cost-effectiveness of the overall system. We also propose a system model and conduct a detailed mathematical analysis. The closed-form solutions we derived can help operators tune parameters without real deployment, saving in cost and time.

The rest of this paper is organized as follows. In Sec. II, we review the related works. In Sec. III and Sec. IV-A, we present the offloading mechanism and the proposed KPG algorithm, respectively. The performance evaluation and experimental results are discussed in Sec. V. Finally, we summarize this paper in Sec. VI.

II. Related Work

Recently, extensive studies have been conducted to investigate cost-effective offloading policies for 5G MEC [13]–[29]. In general, these studies fall into three categories: the impact of preemption [19]–[24], shared resource competition [13], [14], [25]–[29], and offloading decisions [15]–[18].

(1) Preemption: The authors of [19]–[21] consider homogeneous jobs in edge-cloud offloading, in which jobs are treated with the same priority and are balanced between edges and the cloud for better performance. In the heterogeneous job case, i.e., jobs with priorities, high-priority jobs preempt low-priority ones. The authors of [22] study how to increase the usage of transmission resources between two QoS jobs of different priorities. The authors of [23] develop a hybrid detection framework to improve the accuracy of channel information, where the system can reduce the obstacles caused by low-priority jobs and raise the usage of transmission resources. The authors of [24] propose a noncooperative game based on impatient secondary users. The main purpose in [24] is to effectively allocate the transmission resources to low-priority users.

Although the above studies explore the impact of preemption, they focus mainly on bandwidth resource allocation in radio access networks and do not guarantee QoS for low-priority jobs. In this paper our offloading mechanism is tailored to fit 3GPP R16 MEC and takes both preemption and QoS for jobs of different priority into consideration.

(2) Competition: The current studies regarding edge resource competition fall into two categories: bottom-up [13], [14], [25] and top-down [26]–[29]. In bottom-up schemes, edges compete for shared resources at will and are thus able to make offloading decisions quickly. This non-consensus decision-making process, however, may cause congestion and low performance when shared resources are limited. Hence, many bottom-up schemes leverage

game theory to address resource competition in user-edge offloading. The authors of [25] formulate the multiuser offloading competition problem as a potential game and show that the game admits the finite improvement property and always possesses a Nash equilibrium. The authors of [13] discuss how user equipment (UE) avoids congestion by selecting offloading objects using game theory. Later, the same authors [14] prove the existence of Nash equilibrium in the potential game and discuss how to optimize the cost-effectiveness of the overall system.

To avoid the nonconsensus decision-making process of bottom-up schemes, top-down approaches have centralized control of the total system knowledge and perform resource allocation accordingly. The top-down schemes consolidate system information in the central management unit (e.g., SMF), and control edges based on the analyzed results. For example, the authors of [26], [27] use a nonconvex quadratic program to formulate offloading problems and utilize three-step sequential processes to solve the problem of resource allocation. Moreover, the authors of [28] use a graph-based algorithm to describe the communication between edges and find the best resource allocation method through a large number of iterations. In addition to the rule-based algorithms, the authors of [29] adopt reinforcement learning to establish a self-learning mechanism so that the system can automatically control each device to offload. Although the top-down schemes achieve better system performance, their complexity is much higher than that of the bottom-up algorithms, leading to a long decision-making process. In this paper, we take advantage of both approaches and propose KPG algorithm to obtain optimal edge-cloud offloading strategies in (near) real time.

(3) Offloading precision: In 5G MEC, due to the very large jobs and high operating expenses, slight changes in offloading behavior have significant impacts on system performance. Therefore, compared to traditional UE-edge offloading schemes, edge-cloud offloading strategies are difficult to design. The authors of [15] consider that the workload on each edge is too large to integrate entire jobs as an offloading unit; thus, the proposed offloading strategy is to make judgments for each job. Later, the authors of [16]–[18] discuss offloading strategies within edges (referred to as edge-edge offloading), where the proposed strategies can find the best offloading target and determine the offloading ratio for each edge.

The above studies do not consider precision or efficiency. In our proposed mathematical model, we design a granularity coefficient that allows operators to balance analytical precision and computational complexity based on their needs.

III. System Modeling

An overview of the offloading MEC system is illustrated in Fig. 1, where low-priority QoS jobs in each edge server can be offloaded to a SCS through the user plane function (UPF). A policy control function (PCF) then updates offloading policies according to system information from

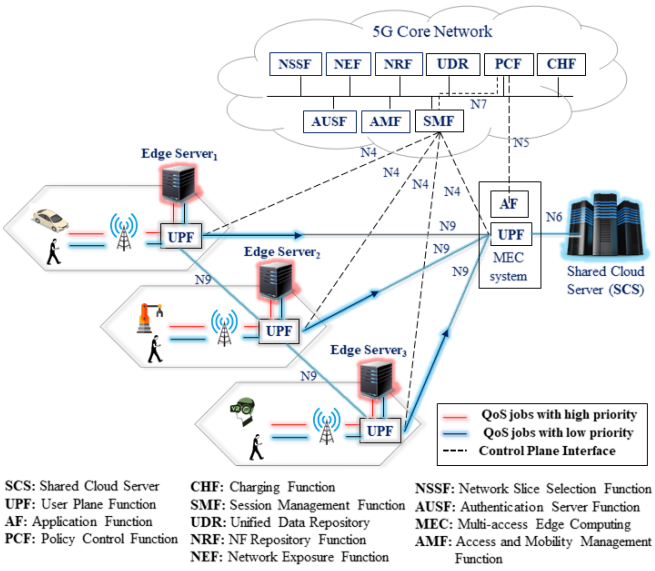


Fig. 1: 5G MEC system in 3GPP Release 16 (R16)

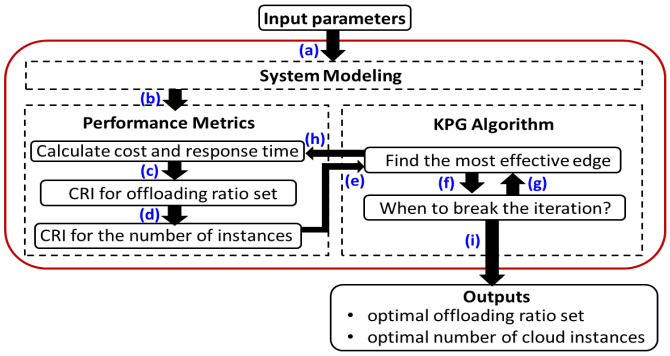


Fig. 2: Input–process–output (IPO) model

an application function (AF) and controls edge-side UPFs through a session management function (SMF). Based on the offloading policies updated by PCF, each UPF in the edge side can offload QoS jobs fractionally to balance the use of shared resources, optimizing the cost-effectiveness of the overall system. In the edge-cloud offloading mechanism, we propose the KPG algorithm to provide a systematic solution. As shown in Fig. 2, our system model consists of the following components:

- Input Parameters: We integrate system information with AF, which includes the traffic, system capabilities, performance bias, cost coefficients, number of edges, and the granularity coefficient.
- System Modeling: By plugging parameters into the system model, we quantify system behaviors in a mathematical way. With the system model, we can test various parameter combinations without actual operation.
- Performance Metrics: To evaluate system performance, we derive closed-form solutions for both response time and costs. We also design a comprehensive metric, CRI, which can be customized according to operators' needs.
- KPG Algorithm: We design KPG algorithm based on

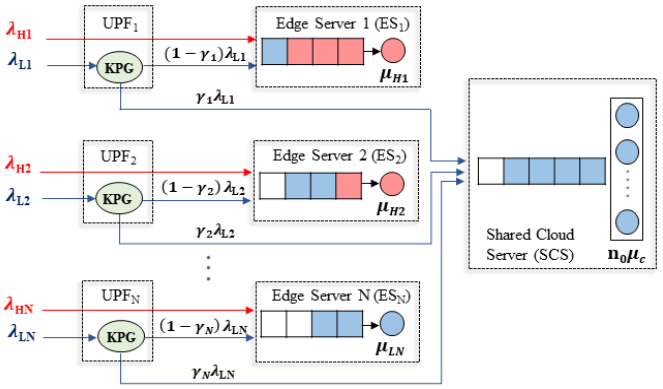


Fig. 3: Queueing model of the 5G MEC system

game theory. Its purpose is to find optimal offloading ratios and the most appropriate number of cloud instances.

- Outputs: According to the KPG algorithm analyzed on PCF, SMF can control UPFs with the optimal offloading ratio set and the number of cloud instances, balancing the cost-effectiveness of the overall system.

As shown in Fig. 2, the whole mechanism consists of 9 procedures. As the first step, procedure (a), we establish a mathematical model to formulate system behaviors based on the given parameters. Next, in procedures (b)-(d), we calculate system performance with closed-form solutions in terms of response time, cost, and CRI. In procedure (e), based on the analysis of the performance metrics, we then find the optimal solutions for both offloading ratio set ($R^*_{n_0}$) and the number of SCS instances (n_0) with the game theory. In procedures (f) and (g), the iterations of the KPG algorithm update the performance of each edge with the change of $R^*_{n_0}$ and n_0. To update the performance metrics, the procedure goes from (h) to procedure (c), which forms the iteration: (c), (d), (e), (f), (g), (h), (c), and so on. Finally, when the CRI of the overall system reaches the global minimum, the optimal solutions of $R^*_{n_0}$ and n_0 can be obtained. In the following sections, we discuss the details of each procedure shown in Fig. 2. The notations are listed in Table I.

As shown in Fig. 3, we consider the 5G MEC system as queueing models consisting of (1) priority queueing models for N numbers of edge servers, and (2) an $M/M/n_0$ queueing model for the SCS. In each edge server, the arrival rate of jobs is divided into λ_{Hi} and λ_{Li} for high-priority and low-priority jobs, respectively. We model the capacity limit of edge i ($i \in \{1, 2, ..., N\}$) as K_i and the average queue lengths of high- and low-priority jobs as l_{Hi} and l_{Li}, respectively. Furthermore, we consider the service rate of high- and low-priority jobs as μ_{Hi} and μ_{Li}, respectively. Each edge offloads jobs to the SCS with offloading ratio γ_i. In the SCS, we consider the number of instances to be n_0, with each instance having a service rate μ_c. The arrival rate of the SCS is λ_c, which is the sum of the low-priority jobs offloaded from the edges.

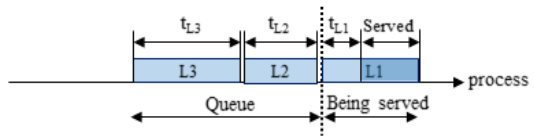
In addition, for the offloading ratio (γ_i, $i \in I$), we take

TABLE I: List of notations

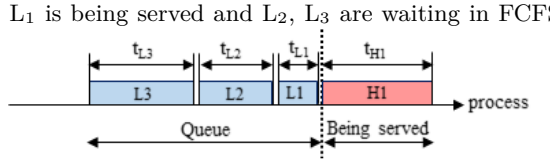
Notation	Definition
$\pi_{n,m}$	The stationary probability for n high-priority jobs and m low-priority jobs.
λ_{Li}	The arrival rate of a low-priority job in edge i
λ_{Hi}	The arrival rate of a high-priority job in edge i
l_{Li}	The queue length of a low-priority job in edge i
l_{Hi}	The queue length of a high-priority job in edge i
K_i	The capacity of edge i
μ_{Li}	The service rate of edge i for a low-priority job
μ_{Hi}	The service rate of edge i for a high-priority job
β_i	The performance bias for setting the offloading ratio of edge i
β_{n_0}	The performance bias for setting the number of instances
c_{beta}	The constant coefficient of the performance bias
γ_i	The offloading ratio of edge i
W_{Hi}	The response time of a high-priority job
W_{Lei}	The response time of a low-priority job in edge i
W_{Li}	The expected value of the response time for a low-priority job through edge i
W_{n_0}	The avg. response time of the entire MEC system when the number of instances is n_0
C_i	The avg. cost caused by edge i
c_{el}	The cost coefficient of low-priority jobs on the edges
c_{eh}	The cost coefficient of high-priority jobs on the edges
c_b	The cost coefficient when an instance is busy
c_i	The cost coefficient when an instance is idle
C_{n_0}	The avg. cost of the entire MEC system when the number of instances is n_0
n_0	The number of instances in the SCS
n_0^*	The optimal number of instances in the SCS
λ_c	The avg. arrival rate in the SCS
μ_c	The service rate of each instance in the SCS
W_c	The avg. response time in the SCS
I	The set of edges
R	The set of offloading ratios
$R^*_{n_0}$	The best offloading ratio set
G	Granularity coefficient: The unit of the offloading ratio
N	The number of edges
ϕ_i	Cost response-time index (CRI): the objective function of edge i
Φ_{R,n_0}	The average value of ϕ_i: the global objective function of the system when the number of instances is n_0
Ψ_{R,n_0}	The objective function of setting the number of instances under the best offloading ratio $R^*_{n_0}$

the impact of precision into consideration. In our system model, we use the granularity coefficient G ($0 < G \leq 1$) as a unit of the offloading ratio. Specifically, there are thousands or more QoS flows within a PDU session between the radio access network and user plane function (UPF). To balance the workload of edge servers, an appropriate offloading ratio (γ) is required to control how many percent of low-priority QoS flows should be offloaded to the back-end SCS. To control the offloading traffic precisely, we introduce granularity coefficient (G) to determine the precision of γ. For example, when $G = 0.5$, the offloading ratio γ can be {0, 0.5, 1}, and when $G = 0.05$, γ can be {0, 0.05, 0.1, 0.15, ..., 1}. If the value of G cannot divide one evenly, the maximum offloading ratio is still 1. Given $G = 0.3$, for example, the offloading ratio can be {0, 0.3, 0.6, 0.9, 1}. Here, we consider the value of G as the input and pay more attention to developing a flexible model that enables operators to analyze offloading ratios according to their requirements.

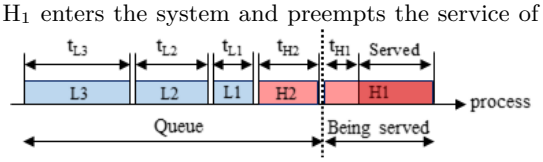
According to [12], real-time preemption mechanism (PREEMPT_RT) can be illustrated as that shown in



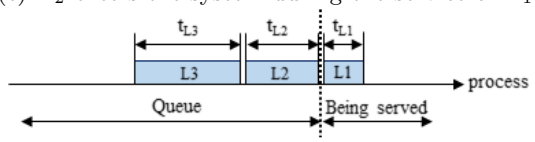
(a) L_1 is being served and L_2, L_3 are waiting in FCFS



(b) H_1 enters the system and preempts the service of L_1



(c) H_2 enters the system during the service of H_1



(d) L_1 continues serving when H_2 finishes

Fig. 4: The process of the PREEMPT_RT on an edge server

Fig. 4. Specifically, same priority jobs follow first come first served (FCFS) rule. High-priority jobs can preempt low-priority jobs. For example, when low-priority job L_1 is being served, L_2 and L_3 must wait for service based on FCFS. However, as shown in Fig. 4(b), when high-priority job H_1 enters the edge, low-priority job L_1 which is being served will be preempted and inserted back at the front of the queue. Next, as shown in Fig. 4(c), when H_1 is being served, another high-priority QoS job H_2 enters the system. According to FCFS, H_2 will queue after H_1 but insert in front of L_1 due to its priority. In addition, the PREEMPT_RT we consider is preempt-resume mechanism, in which the service of the preempted jobs continues from the breakpoint rather than restarting from the beginning. Thus, as shown in Fig. 4(d), when L_1 is served again, the service will take the remaining time, t_{L1}.

IV. Performance Analysis

A. Performance Metrics

In this section, we formulate the edge behavior with a 2D-Markov chain. As shown in Fig. 5, we define $\pi_{n,m}$ as the stationary probability for the state of an edge server with n high-priority jobs and m low-priority jobs. By solving the balance equations, we can obtain the following performance metrics, laying the cornerstone for the cost-effectiveness optimization in Sec. IV-B.

- Response time of high-priority jobs (W_{Hi}): Because of the PREEMPT-RT mechanism, high-priority jobs can not only preempt the resource on the edge servers but also not be affected by low-priority jobs. The service of high-priority jobs follows the FCFS. Thus, W_{Hi} can be easily derived with one dimension Markov chain.
- Response time of low-priority jobs (W_{Li}): Due to the offloading mechanism, the response time of low-priority jobs is defined as the expectation value from the edges (W_{Lei}) and the SCS (W_c).
- Response time of the SCS (W_c): In the 5G MEC system, the response time of the SCS not only affects the offloading ratios of the edges but also results in a trade-off problem with the cost of deployment.
- The cost of an edge (C_i): The cost considered in this paper consists of the number of jobs served by the system and the number of instances set up in the back-end SCS.
- Cost response-time index (CRI): CRI is a comprehensive metric widely used to address cost-effectiveness. In CRI, two weight factors are introduced, enabling operators to customize their objective functions based on their system bias. In this paper, we use CRI to optimize the system's cost-effectiveness.

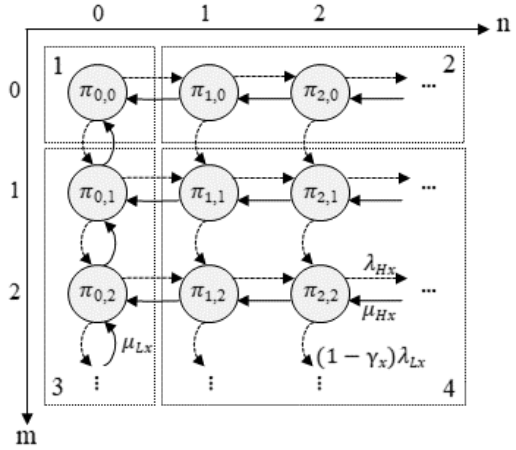


Fig. 5: 2D-Markov chain of priority queueing system

Here, we formally quantify the above performance metrics. First, we separate the balance equations into four parts:

(Part 1) $m = 0$, $n = 0$: Part 1 indicates that there are neither high- nor low-priority jobs accessing the edge server i.

$$\frac{d\pi_{0,0}^t}{dt} = -\left[\lambda_{Hi} + (1-\gamma_i)\,\lambda_{Li}\right]\pi_{0,0}^t + \mu_{Hi}\pi_{1,0}^t + \mu_{Li}\pi_{0,1}^t \tag{1}$$

(Part 2) $m = 0$, $n > 0$: Part 2 means that only high-priority jobs are accessing the edge server i.

$$\frac{d\pi_{n,0}^t}{dt} = -\left[\lambda_{Hi} + (1-\gamma_i)\,\lambda_{Li} + \mu_{Hi}\right]\pi_{n,0}^t + \lambda_{Hi}\pi_{n-1,0}^t + \mu_{Hi}\pi_{n+1,0}^t \tag{2}$$

(Part 3) $m > 0$, $n = 0$: Part 3 means that only low-priority

jobs access the edge server i.

$$\frac{d\pi^t_{0,m}}{dt} = -\left[\lambda_{Hi} + (1-\gamma_i)\,\lambda_{Li} + \mu_{Li}\right]\pi^t_{0,m} + (1-\gamma_i)\,\lambda_{Li}\pi^t_{0,m-1} + \mu_{Li}\pi^t_{0,m+1} + \mu_{Hi}\pi^t_{1,m} \quad (3)$$

(Part 4) $m > 0$, $n > 0$: Part 4 shows that both high- and low-priority jobs access the edge server i.

$$\frac{d\pi^t_{n,m}}{dt} = -\left[\lambda_{Hi} + (1-\gamma_i)\,\lambda_{Li} + \mu_{Hi}\right]\pi^t_{n,m} + \lambda_{Hi}\pi^t_{n-1,m} + \mu_{Hi}\pi^t_{n+1,m} + (1-\gamma_i)\,\lambda_{Li}\pi^t_{n,m-1}. \quad (4)$$

The above probability states should satisfy the normalization condition as follows:

$$\sum_{n=0}^{\infty}\sum_{m=0}^{\infty}\pi^t_{n,m} = 1. \quad (5)$$

Furthermore, in the steady state, the differential terms of time must be zero:

$$\frac{d\pi^t_{0,0}}{dt} = \frac{d\pi^t_{n,0}}{dt} = \frac{d\pi^t_{0,m}}{dt} = \frac{d\pi^t_{n,m}}{dt} = 0. \quad (6)$$

According to the PREEMPT_RT scheduling mechanism, both edges and the SCS impact low-priority jobs. The expected value of the response time for low-priority job W_{Li} can be expressed as:

$$W_{Li} = (1-\gamma_i)W_{Lei} + \gamma_i W_c. \quad (7)$$

Then, we can obtain W_{Lei}, the response time of low-priority jobs served by edge i:

$$W_{Lei} = l_{Li}/[(1-\gamma_i)\lambda_{Li}]. \quad (8)$$

Since high-priority jobs are directly solved at the edge side, we obtain the average system response time of high-priority jobs as (9):

$$W_{Hi} = l_{Hi}/\lambda_{Hi}. \quad (9)$$

A limited buffer size means that excessive arrival jobs will be dropped. In this paper, however, we consider the offloading mechanism when the arrival rate gets close to the service rate on the edge server, and UPF will offload an appropriate number of low-priority jobs to the back-end SCS to reduce the workload of the edge servers. Thus, for the near-end edge servers, we do not specifically limit the buffer size because the excessive number of low-priority jobs will be offloaded to the back-end SCS. On the other hand, although the back-end SCS can be scaled dynamically, the cost will be a concern. Therefore, even if the resource of SCS can be extended, we still treat it as a server with limited capacity. In this way, we can mitigate the workload on the near-end edge servers and reduce the cost of the back-end SCS.

In the following paragraphs, we expound in detail about the derivations of the closed-form solutions and show the results of l_{Li} and l_{Hi} as below:

(1) The average number of low-priority jobs in edge i

By plugging (1), (2), (3), (4) into (10), we can convert the 2D-Markov chain to one dimension and separate the derivations into *Part A* ($n > 0$) and *Part B* ($n = 0$) as follows:

$$H_n(z) = \sum_{m=0}^{\infty}\pi_{n,m}z^m, \qquad z \in [-1,1], \quad (10)$$

Part A ($n > 0$): combine (2) and (4) in (10)

$$\begin{aligned} 0 = &-\left[\lambda_{Hi} + (1-\gamma_i)\,\lambda_{Li} + \mu_{Hi}\right]\pi^t_{n,0}z^0 + \lambda_{Hi}\pi^t_{n-1,0}z^0 \\ &+ \mu_{Hi}\pi^t_{n+1,0}z^0 \\ &+ \sum_{m=1}^{\infty} -\left[\lambda_{Hi} + (1-\gamma_i)\,\lambda_{Li} + \mu_{Hi}\right]\pi^t_{n,m}z^m \\ &+ \lambda_{Hi}\pi^t_{n-1,m}z^m + \mu_{Hi}\pi^t_{n+1,m}z^m \\ &+ (1-\gamma_i)\,\lambda_{Li}\pi^t_{n,m-1}z^m. \end{aligned} \quad (11)$$

Next, we can reorganize (11) in terms of $H_n(z)$ as follows:

$$\begin{aligned} &-\left[\lambda_{Hi} + (1-\gamma_i)\,\lambda_{Li} + \mu_{Hi}\right]H_n(z) = \\ &-\left[\lambda_{Hi} + (1-\gamma_i)\,\lambda_{Li} + \mu_{Hi}\right]\pi^t_{n,0}z^0 \\ &+ \sum_{m=1}^{\infty} -\left[\lambda_{Hi} + (1-\gamma_i)\,\lambda_{Li} + \mu_{Hi}\right]\pi^t_{n,m}z^m \end{aligned} \quad (12)$$

$$\lambda_{Hi}H_{n-1}(z) = \lambda_{Hi}\pi^t_{n-1,0}z^0 + \sum_{m=1}^{\infty}\lambda_{Hi}\pi^t_{n-1,m}z^m \quad (13)$$

$$\mu_{Hi}H_{n+1}(z) = \mu_{Hi}\pi^t_{n+1,0}z^0 + \sum_{m=1}^{\infty}\mu_{Hi}\pi^t_{n+1,m}z^m \quad (14)$$

$$(1-\gamma_i)\,\lambda_{Li}\left[zH_n(z)\right] = \sum_{m=1}^{\infty}(1-\gamma_i)\,\lambda_{Li}\pi^t_{n,m-1}z^m. \quad (15)$$

Finally, through (12), (13), (14), and (15), we can convert (11) into (16):

$$\begin{aligned} 0 = &\,\mu_{Hi}H_{n+1}(z) \\ &-\left[\lambda_{Hi} + (1-z)\,(1-\gamma_i)\,\lambda_{Li} + \mu_{Hi}\right]H_n(z) \\ &+ \lambda_{Hi}H_{n-1}(z). \end{aligned} \quad (16)$$

As quadratic functions, we denote $\xi(z)$ as the roots of (16):

$$\xi(z) = \frac{\left[\lambda_{Hi} + (1-z)\,(1-\gamma_i)\,\lambda_{Li} + \mu_{Hi}\right]}{2\mu_{Hi}} \pm \frac{\sqrt{\left[\lambda_{Hi} + (1-z)\,(1-\gamma_i)\,\lambda_{Li} + \mu_{Hi}\right]^2 - 4\lambda_{Hi}\mu_{Hi}}}{2\mu_{Hi}} \quad (17)$$

Nevertheless, only "$-$" one can be selected due to the limit $z \in [-1,1]$. We can therefore convert the 2D-Markov chain to one dimension which follows the rule:

$$H_n(z) = \xi^n(z)H_0(z), \qquad n > 0. \quad (18)$$

Part B ($n = 0$): combine (1) and (3) in (10)

$$\begin{aligned} 0 = &\\ &-\left[\lambda_{Hi} + (1-\gamma_i)\,\lambda_{Li}\right]\pi^t_{0,0}z^0 + \mu_{Hi}\pi^t_{1,0}z^0 + \mu_{Li}\pi^t_{0,1}z^0 \\ &+ \sum_{m=1}^{\infty} -\left[\lambda_{Hi} + (1-\gamma_i)\,\lambda_{Li} + \mu_{Li}\right]\pi^t_{0,m}z^m \\ &+ (1-\gamma_i)\,\lambda_{Li}\pi^t_{0,m-1}z^m + \mu_{Li}\pi^t_{0,m+1}z^m + \mu_{Hi}\pi^t_{1,m}z^n \end{aligned} \quad (19)$$

To merge the formula according to the same n, we extend (19) as follows:

$$\begin{aligned}0 =& -\left[\lambda_{Hi}+(1-\gamma_i)\,\lambda_{Li}+\mu_{Li}\right]\pi^t_{0,0}z^0+\mu_{Li}\pi^t_{0,0}z^0\\&+\mu_{Hi}\pi^t_{1,0}z^0+\mu_{Li}\frac{1}{z}\pi^t_{0,1}z^1+\mu_{Li}\frac{1}{z}\pi_{0,0}z^0-\mu_{Li}\frac{1}{z}\pi_{0,0}z^0\\&+\sum_{m=1}^{\infty}-\left[\lambda_{Hi}+(1-\gamma_i)\,\lambda_{Li}+\mu_{Li}\right]\pi^t_{0,m}z^m\\&(1-\gamma_i)\,\lambda_{Li}\pi^t_{0,m-1}z^m+\mu_{Li}\pi^t_{0,m+1}z^m+\mu_{Hi}\pi^t_{1,m}z^n.\end{aligned}\tag{20}$$

Next, we reorganize (20) in terms of $H_n(z)$ as follows:

$$\begin{aligned}&-\left[\lambda_{Hi}+(1-\gamma_i)\,\lambda_{Li}+\mu_{Li}\right]H_0(z)=\\&-\left[\lambda_{Hi}+(1-\gamma_i)\,\lambda_{Li}+\mu_{Li}\right]\pi^t_{0,0}z^0\\&+\sum_{m=1}^{\infty}-\left[\lambda_{Hi}+(1-\gamma_i)\,\lambda_{Li}+\mu_{Li}\right]\pi^t_{0,m}z^m,\end{aligned}\tag{21}$$

$$\mu_{Hi}H_1(z)=\mu_{Hi}\pi^t_{1,0}z^0+\sum_{m=1}^{\infty}\mu_{Hi}\pi^t_{1,m}z^n,\tag{22}$$

$$\lambda_{Li}\left[zH_0(z)\right]=\sum_{m=1}^{\infty}\lambda_{Li}\pi^t_{0,m-1}z^m,\tag{23}$$

$$\begin{aligned}&\mu_{Li}\frac{1}{z}H_0(z)=\mu_{Li}\frac{1}{z}\pi_{0,0}z^0+\mu_{Li}\frac{1}{z}\pi^t_{0,1}z^1\\&+\sum_{m=1}^{\infty}\mu_{Li}\frac{1}{z}\pi^t_{0,m+1}z^{m+1}.\end{aligned}\tag{24}$$

Finally, through (21), (22), (23), and (24), we convert (19) into (25):

$$\begin{aligned}0=&-\left[\lambda_{Hi}+(1-\gamma_i)\,(1-z)\,\lambda_{Li}+\left(1-\frac{1}{z}\right)\mu_{Li}\right]H_0(z)\\&+\mu_{Hi}H_1(z)+\left(1-\frac{1}{z}\right)\mu_{Li}\pi^t_{0,0},\qquad n=0.\end{aligned}\tag{25}$$

By combining (16) and (25), we can derive $H_0(z)$ as follows:

$$\begin{aligned}&H_0(z)\\&=\frac{\mu_{Li}\left(1-\frac{1}{z}\right)\pi^t_{0,0}}{\left[\mu_{Li}\xi(z)-\lambda_{Hi}-(1-\gamma_i)\,(1-z)\,\lambda_{Li}+\left(1-\frac{1}{z}\right)\mu_{Li}\right]}.\end{aligned}\tag{26}$$

With (26), we can calculate the average number of low-priority jobs in the system by partially differentiating $H_n(z)$, which is shown as follows:

$$\begin{aligned}&l_{Li}=\sum_{n=0}^{\infty}\frac{\partial H_n(z)}{\partial z}|_{z=1}=\frac{(\frac{1}{\mu_{Li}})[1+(\frac{\mu_{Li}}{\mu_{Hi}})(\frac{\rho_{Hi}}{\rho_{Li}})]}{1-\rho_{Hi}-\rho_{Li}},\\&\rho_{Hi}=\frac{\lambda_{Hi}}{\mu_{Hi}},\ \text{and}\ \rho_{Li}=\frac{(1-\gamma_i)\lambda_{Li}}{\mu_{Li}}.\end{aligned}\tag{27}$$

(2) The average number of high-priority jobs in edge i
As in (10), we use the z-transform to derive the average number of high-priority jobs and use (28) as the PGF corresponding to m low-priority jobs:

$$P_m(z)=\sum_{n=0}^{\infty}\pi_{n,m}z^n,\qquad z\in[-1,1].\tag{28}$$

However, because high-priority jobs are not affected by jobs with low priority, $P_m(z)$ is the same for $m=\{0,1,2...\}$, and the balance equations can be rewritten as follows:

$$\left(\lambda_{Hi}+\mu_{Hi}\right)\pi^t_{n,m}=\lambda_{Hi}\pi^t_{n-1,m}+\mu_{Hi}\pi^t_{n+1,m},\ n>0.\tag{29}$$

$$\lambda_{Hi}\pi^t_{0,m}=\mu_{Hi}\pi^t_{1,m}.\tag{30}$$

By plugging (29) into (28), we obtain the following equation:

$$\begin{aligned}&\sum_{n=1}^{\infty}\pi^t_{n+1,m}z^n=(\rho_{Hi}+1)\sum_{n=1}^{\infty}\pi^t_{n,m}z^n-\rho_{Hi}\sum_{n=1}^{\infty}\pi^t_{n-1,m}z^n\\&,\ \rho_{Hi}=\frac{\lambda_{Hi}}{\mu_{Hi}},\end{aligned}\tag{31}$$

which can be presented in terms of $P_m(z)$:

$$\begin{aligned}&\frac{1}{z}\left[P_m(z)-\pi_{1,m}(z)z^1-\pi_{0,m}\right]=(\rho_{Hi}+1)[P_m(z)-\pi_{0,m}]\\&-(\rho_{Hi})[zP_m(z)].\end{aligned}\tag{32}$$

From (32), we can further derive $P_m(z)$ as:

$$P_m(z)=\frac{\pi_{0,m}}{1-\rho_{Hi}z},\tag{33}$$

in which we find that $\pi_{0,m}=(1-\rho_{Hi})$ with the feature of the PGF ($P_m(1)=1$). We can therefore obtain $\pi_{n,m}=(1-\rho_{Hi})\rho^n_{Hi}$ by inserting $\pi_{0,m}$ back into (29) and (30). Finally, we can find the mean number of high-priority jobs as follows:

$$l_{Hi}=\sum_{n=0}^{\infty}n\times\pi_{n,m}=\frac{\rho_{Hi}}{1-\rho_{Hi}}.\tag{34}$$

On the other hand, we test multiple queuing models to quantify the behaviors of the SCS; however, the behavior does not differ substantially between each probability distribution. To describe the behavior of the SCS, we therefore adopt a classic queueing model, i.e., M/M/n_0, where the average response time (W_c) is shown in (35).

$$\begin{aligned}&W_c=\frac{\frac{1}{(1-\rho_c)(n_0\mu_c)}}{1+(1-\rho_c)[\frac{n_0!}{(n_0\rho_c)^{n_0}}][\sum_{k=0}^{n_0-1}\frac{(n_0\rho_c)^k}{k!}]}+\frac{1}{\mu_c},\\&\rho_c=\frac{\lambda_c}{n_0\mu_c}\ ,\ \lambda_c=\sum_{i=1}^{N}\gamma_i\times\lambda_{Li}.\end{aligned}\tag{35}$$

In this paper, we quantify the cost with the number of tasks and instances. Specifically, we consider the tasks on the edge side as high and low priorities and denote their cost coefficients as c_{eh} and c_{el}, respectively. For the SCS, we denote c_b as the cost coefficient of instances in busy state, c_i for that in idle state, and (n_b, n_i) for the numbers of instances in busy and idle states, respectively. The proportion of the cost coefficients can be tuned by

operators based on their needs. In the objective function CRI (Sec. IV-A1), both the cost and response time are normalized. In this way, operators can determine the cost coefficients without constraints. Finally, we use C_i to represent the costs arising from edge $i \in I$, which is derived as follows:

$$C_i = c_{eh}\lambda_{Hi} + c_{el}(1-\gamma_i)\lambda_{Li} + (c_b n_b + c_i n_i)\gamma_i \lambda_{Li}. \quad (36)$$

To quantify the cost-effectiveness, we introduce CRI [30], which is a comprehensive metric of both cost and response time. In CRI, we consider the bias weighting factors (β_{resp} and β_{cost}), which can enable operators to balance CRI between cost and performance based on their needs. In this section, we formulate the analysis of CRI for the offloading ratio set ($R = \{\gamma_i, i \in I\}$) and number of instances (n_0) in Sec. IV-A1 and Sec. IV-A2, respectively. We prove the convexity of the CRI in Sec. IV-B and evaluate the performance based on CRI in Sec. V.

1) CRI for the offloading ratio set (Φ_{R,n_0}): To analyze the best offloading ratio for each edge, we first specify CRI value of edge i with ϕ_i:

$$\phi_i = \beta_{resp} \times Norm\{W_{Li}\} + \beta_{cost} \times Norm\{C_i\}, \quad (37)$$

where

$$Norm\{W_{Li}\} = W_{Li}/W_{Li\_max},\ W_{Li\_max} = (W_{Li}|\gamma_i = 0),$$
$$Norm\{C_i\} = C_i/C_{i\_max},\ C_{i\_max} = (C_i|\gamma_i = 1).$$

W_{Li_max} is the response time when $\gamma_i = 0$, which means that edge i does not offload low-priority jobs to SCS. Conversely, C_{i_max} is the cost when $\gamma_i = 1$, which indicates that the edge i totally offloads its low-priority jobs to the SCS. In addition, we design bias weighting factors (β_{resp} and β_{cost}) as (38):

$$\beta_{resp} = \frac{1}{c_{beta} \times (1-\beta_i)},\ \beta_{cost} = \frac{1}{c_{beta} \times \beta_i}. \quad (38)$$

Operators can simultaneously adjust both β_{resp} and β_{cost} by changing the performance bias $0 < \beta_i < 1$ and making the magnitude uniform with the constant c_{beta}. The advantage of (38) is that it turns (37) into a convex function in terms of β_i, such that operators can specify CRI for each edge according to its cost-effective requirements. By multiplying the bias weights, we can formulate CRI for each edge and denote the objective function of the overall system as Φ_{R,n_0}. As shown in (39), the major challenge of this paper is to find the best offloading ratio set ($R^*_{n_0}$) that can reach the minimum point of (39):

$$\underset{R=\{\gamma_i,\ i\in I\}}{\operatorname{argmin}} (\Phi_{R,n_0} = \frac{1}{N}\sum_{i=1}^{N}\phi_i). \quad (39)$$

2) CRI for the number of instances (Ψ_{n_0}): To analyze the most appropriate number of instances for the SCS, we specify CRI value in terms of the normalized response time and cost:

$$\Psi_{n_0} = \beta^{'}_{resp} \times Norm\{W_{n_0}\} + \beta^{'}_{cost} \times Norm\{C_{n_0}\}. \quad (40)$$

Algorithm 1: Knapsack Potential Game (KPG)

Input: from edges:
$\{\lambda_{Li}, \lambda_{Hi}, \mu_{Li}, \mu_{Hi}, \beta_i, c_{el}, c_{eh} | i \in I\}$;
from cloud: $\{n_0, \mu_c, c_b, c_i\}$; others: $\{N, G\}$
Output: Offloading ratio set
$R^*_{n_0} = \{r^*_1, r^*_2, ..., r^*_i | i \in I\}$

Let MEC to be a vector of the Edge set;
Let SCS to be a Cloud set;
Let edge_list to be a vector of integers;
/* Initialize */
$MEC.reserve(N)$; $edge\_list.reserve(N)$;
for i (*0 to* $N-1$) do
  Let E to be an Edge set;
  $E \leftarrow \{\lambda_{Li}, \lambda_{Hi}, \mu_{Li}, \mu_{Hi}, 0, 0, 0, c_{el}, c_{eh}, c_b, c_i, 1, 0, \beta_i\}$
  $MEC.push\_back(E)$;
  $edge\_list.push\_back(i)$;
end
$SCS \leftarrow \{n_0, \mu_c, 0, 0\}$
Bagging($MEC, SCS, edge\_list$) ; // Algorithm 2
return offloading ratio set
$R^*_{n_0} = (MEC[i].\gamma \mid i \in I)$;

Different from (37), both the normalized response time and cost considered here are the average values of the overall system:

$$Norm\{W_{n_0}\} = W_{n_0}/W_{n_0\_max}, W_{n_0\_max} = (W_{n_0}|n_0 = 0),$$
$$Norm\{C_{n_0}\} = C_{n_0}/C_{n_0\_max}, C_{n_0\_max} = (C_{n_0}|n_0 = max),$$

where

$$W_{n_0} = \frac{1}{N}\sum_{i=1}^{N} W_{L_i}, \quad C_{n_0} = \frac{1}{N}\sum_{i=1}^{N} C_i. \quad (41)$$

To increase the flexibility of the analysis, we specifically design additional bias weighting factors ($\beta^{'}_{resp}$ and $\beta^{'}_{cost}$) according to (42), where β_{n_0} is different from β_i in (38). β_i allows operators to set a different performance bias for each edge, whereas β_{n_0} enables operators to set the performance bias when they need to determine the number of instances on the SCS. By tuning β_{n_0}, operators can simultaneously control both bias weighting factors according to their requirements:

$$\beta^{'}_{resp} = \frac{1}{c_{beta} \times (1-\beta_{n_0})},\ \beta^{'}_{cost} = \frac{1}{c_{beta} \times \beta_{n_0}}. \quad (42)$$

Based on the analysis of the optimal offloading ratio set ($R^*_{n_0}$) for each n_0, we can find the optimal number of instances (n^*_0) to minimize (43):

$$\underset{n_0}{\operatorname{argmin}}(\Psi_{R,n_0}). \quad (43)$$

B. Knapsack Potential Game (KPG): The Proposed Algorithm to Find $R^*_{n_0}$

We propose the KPG algorithm to analyze the optimal offloading ratio for edges and achieve optimal cost-effectiveness for the 5G MEC system. In the KPG algorithm, we combine the knapsack problem and potential

Algorithm 2: Bagging

Input: MEC, SCS, $edge\_list$
Output: $Integer$ /* $1 : Complete$ */
$Set\ \phi_{now},\ \phi_{next},\ \phi_{diff},\ \Phi_{before},\ \Phi_{after}\ to\ 0$;
while 1 do
  /* $Find\ most\ value\ edge$ */;
  $move \leftarrow 0$;
  $\phi_{diff} \leftarrow 0$;
  for i ($0\ to\ edge\_list.size() - 1$) do
    $j \leftarrow edge\_list[i]$;
    $\phi_{now} \leftarrow \{(37) \mid \gamma_i = MEC[j].\gamma\}$;
    $\phi_{next} \leftarrow \{(37) \mid \gamma_i = MEC[j].\gamma + G\}$;
    if $\{(45) \mid {}_{\triangle}\phi = \phi_{next} - \phi_{now}\} > \phi_{diff}$ then
      $move \leftarrow j$;
      $\phi_{diff} \leftarrow \{(45) \mid {}_{\triangle}\phi = \phi_{next} - \phi_{now}\}$;
    else if $\{(45) \mid {}_{\triangle}\phi = \phi_{next} - \phi_{now}\} \leq 0$ then
      /* $Pruning$ */;
      $edge\_list.erase(edge\_list.begin() + i)$;
    end
  end
  /* $Update\ global\ phi$ */;
  $\Phi_{before} \leftarrow (39)$;
  $MEC[move].\gamma \leftarrow (MEC[move].\gamma + G)$;
  $\Phi_{after} \leftarrow (39)$;
  if $\Phi_{after} \geq \Phi_{before}$ then
    /* $Last\ one\ should\ not\ be\ counted$ */;
    $MEC[move].\gamma \leftarrow (MEC[move].\gamma - G)$;
    return 1;
  end
end

game to improve efficiency and performance, respectively. Further, we consider the following factors:

1) Knapsack Problem:
Knapsack problem [31] is a well-known NP-complete problem. To find the maximum-weight composition of the items within a size-limited bag, the greedy method is a common approach to solve the problem. Specifically, we need to put items into the knapsack in the order of cost-effectiveness. Therefore, a straightforward way is to calculate the $weight/size$ of each item and put the items into the knapsack according to the sorted result. However, this method is not suitable to solve the problem of edge-cloud offloading. In the edge-cloud offloading problem, we treat low-priority QoS flows as the items, the response time as the weight, and the capacity of the SCS as the knapsack. We try to appropriately offload low-priority jobs to the back-end SCS to optimize system response time. However, a fully loaded back-end SCS will cause system performance degradation. Thus, the knapsack problem still needs an objective function to balance the resource allocation of the SCS among the edge servers.
2) Potential Game:
In traditional game theory, multiple players are in a game, and each player gets benefits according to their choices. Different from the traditional game theory, however, a potential game has a global objective function, which can be fell into many kinds of potential games (e.g., exact potential game, weight potential game, ordinal potential game, best response potential game). In KPG, we refer to the best response potential game (BRPG) [32] because the actions made by each player in BRPG must be the best choice for the overall system. Nevertheless, the efficiency of the BRPG will be another concern because BRPG needs to calculate the objective function for each player's action. To this end, we design the KPG to obtain the same performance as BRPG but improve efficiency. For further verification, please refer to Sec. V.
3) Queuing model:
In the optimization process, we need to calculate the results for a large number of parameter settings. A queuing model is indispensable to compute the result for each set of parameters quickly. For example, if there are N edge servers and each of them has 10 offloading selections (s.e., 0.1, 0.2, ..., 1), there will be 10^N different results. To this end, the major purpose of the KPG algorithm is to prune out the irrational settings, and the queuing model is to calculate the results for the sets that KPG needs.

In Algorithm 1, we denote the parameters used on the edges and SCS as the sets Edge: $\{\lambda_L,\ \lambda_H,\ \mu_L,\ \mu_H,\ \phi,\ W_L,\ C,\ c_{el},\ c_{eh},\ c_b,\ c_i,\ \gamma,\ \beta\}$, $Edge \subset MEC$ and Cloud: $\{n_0, \mu_c, \lambda_c, W_c\}$, respectively. Then, KPG is designed based on the best response potential game (BRPG) [32], where all players' decisions are the best response to the overall system. As shown in Algorithm 1, we consider each Edge $i \in I$ as a player in the game, and formulize players' behaviours as the fractional knapsack problem (FKP) [31]. That is, a player's strategy is either to maintain the current status or to increase the offloading ratio. As each player may leave the game at any time, we use $edge\_list$ to record the players that remain in the game. After the initialization, Algorithm 2 is called to calculate and return the optimized offloading ratios.

In Function 2, we use (37) to denote the current self-benefit of a player as ϕ_{now} and denote the corresponding self-benefit as ϕ_{next} when a player selects the strategy $increase$. In addition, we use (39) to represent the utility of the overall system as Φ and separate it into Φ_{before} and Φ_{after} to observe the impact on system performance when a player adjusts their strategy. In the following paragraphs, we list the major challenges of the optimization of edge-cloud offloading and give details regarding how to use the iterations of KPG to find the best offloading ratio with the time complexity of $O(N^2)$.

1) High dependency in edges: In the 5G MEC system, the edges are highly related. Even a slight adjustment of the offloading ratio of one edge will have a significant impact on system performance. In our proposed KPG, when a player (edge) changes strategy (remains/increases

the offloading ratio), the utilities (37) of all players are affected. Therefore, to ensure that the iterative process is always being optimized, in each iteration, only the most effective edge is allowed to increase the offloading ratio by a unit (G). However, the accuracy requirement constrains the analysis duration. Here, we design a pruning mechanism to improve the efficiency of KPG analysis. Because the utility obtained by increasing the offloading ratio decreases as SCS utilization increases, the trend of (37) is a convex function in terms of the offloading ratio. Thus, Function 2 can remove an edge when it has no incentive to increase its offloading ratio ($-_{\triangle}\phi_i \leq 0$). By pruning redundant players in the game, we can accelerate the analysis without affecting the results.

2) How to find the most effective edge: As mentioned above, due to the high dependency, the system should find the most effective edge to raise the overall system utility in each iteration. However, the self-benefit of each edge continues to change during KPG iterations, so the most effective edge is not always the same. To address this problem, BRPG can find the most effective edge by comparing the change in overall system utility (as shown in (44)). However, even if BRPG guarantees that the edge it selected is the most effective, the time complexity will be increased substantially. To address this problem, we reference the solution of FKP [31], which increases edges' offloading ratios in a cost-effective manner. As shown in (45), FKP needs to calculate only the change in (37) rather than that in (39) because the solutions of (44) and (45) will be the same ($i = i'$). In Function 2, we use the variable *move* to record the most effective edge and increase its offloading ratio by a unit (G). In the following paragraph, we offer more details regarding the equivalence ($i = i'$) between (44) and (45):

$$\underset{i\in I}{\operatorname{argmax}}(S = -\frac{\triangle\Phi}{\triangle\gamma_i \times \lambda_{Li}}), \tag{44}$$

$$\underset{i'\in I}{\operatorname{argmax}}(s = -\frac{\triangle\phi_{i'}}{\triangle\gamma_{i'} \times \lambda_{Li'}}). \tag{45}$$

For (44), we denote the values of the edges i' and j' as $S_{i'}$ and $S_{j'}$, respectively. Similarly, in (45), $s_{i'}$ and $s_{j'}$ denote the values of edges i' and j', respectively. Here, we assume that i' is the solution of (45); that is, $s_{i'}$ is the largest value in all $s_k, k \in I$. As aforementioned discussion, when i' is the solution of (45), it must be the solution of (44), which means that $S_{i'}$ is also the largest value in all $S_k, k \in I$. Here, we use the other edge j' to conduct the proof in contradiction, where we assume that $s_{j'} < s_{i'}$ but $S_{j'} > S_{i'}$. We can confirm the equivalence between (44) and (45) by proving the irrationality of edge j'.

Proof: Since the forms of $S_{i'}$ and $S_{j'}$ are fractions, we can discuss $s_{j'} < s_{i'}$ in terms of two cases: $\{|_{\triangle}\phi_{j'}| = |_{\triangle}\phi_{i'}|,\ _{\triangle}\gamma_{j'}\lambda_{Lj'} >_{\triangle} \gamma_{i'}\lambda_{Li'}\}$ and $\{_{\triangle}\gamma_{j'}\lambda_{Lj'} =_{\triangle} \gamma_{i'}\lambda_{Li'},\ |_{\triangle}\phi_{j'}| < |_{\triangle}\phi_{i'}|\}$. According to (39) and Function 2, when an edge is allowed to increase the offloading ratio by a unit ($1/G$), the objective functions of the other edges will increase due to the growth in the SCS response time (w_c). Here, we denote the sum of the increases in the objective functions by edge i' as $_{\triangle}\Phi_{\bar{i'}}$ and that by edge j' as $_{\triangle}\Phi_{\bar{j'}}$ so that the change in the global objective function caused by edges i' and j' can be represented as $_{\triangle}\Phi_{i'} =_{\triangle} \Phi_{\bar{i'}} +_{\triangle} \phi_{i'}$ and $_{\triangle}\Phi_{j'} =_{\triangle} \Phi_{\bar{j'}} +_{\triangle} \phi_{j'}$, respectively. In addition, because the iteration will not terminate until $_{\triangle}\Phi \geq 0$, the values of $_{\triangle}\Phi_{i'}$, $_{\triangle}\phi_{i'}$, $_{\triangle}\Phi_{j'}$ and $_{\triangle}\phi_{j'}$ should be negative, but those of $_{\triangle}\Phi_{\bar{i'}}$ and $_{\triangle}\Phi_{\bar{j'}}$ must be positive during the iterations. Therefore, in the first case, i.e., $\{|_{\triangle}\phi_{j'}| = |_{\triangle}\phi_{i'}|,\ _{\triangle}\gamma_{j'}\lambda_{Lj'} >_{\triangle} \gamma_{i'}\lambda_{Li'}\}$, $_{\triangle}\Phi_{\bar{j'}}$ will be larger than $_{\triangle}\Phi_{\bar{i'}}$, which causes the result of (44) to be $S_{j'} < S_{i'}$ because $(-_{\triangle}\Phi_{j'}) < (-_{\triangle}\Phi_{i'})$ with the denominators $_{\triangle}\gamma_{j'}\lambda_{Lj'} >_{\triangle} \gamma_{i'}\lambda_{Li'}$. For the other case, i.e., $\{_{\triangle}\gamma_{j'}\lambda_{Lj'} =_{\triangle} \gamma_{i'}\lambda_{Li'},\ |_{\triangle}\phi_{j'}| < |_{\triangle}\phi_{i'}|\}$, even though $_{\triangle}\Phi_{\bar{j'}}$ is equal to $_{\triangle}\Phi_{\bar{i'}}$, $|_{\triangle}\phi_{j'}| < |_{\triangle}\phi_{i'}|$ will also cause the result of (44) to be $S_{j'} < S_{i'}$ because $(-_{\triangle}\Phi_{j'}) < (-_{\triangle}\Phi_{i'})$ with the same denominators ($_{\triangle}\gamma_{j'}\lambda_{Lj'} =_{\triangle} \gamma_{i'}\lambda_{Li'}$). Finally, by proving the irrationality of edge j' with the above two cases, we can confirm the equivalence between (44) and (45), which means that when edge i' is the solution of (45), it must be the solution of (44) as well.

3) When to break the iteration: Unlike the solution of FKP, we do not entirely fill the SCS at the end of the iteration. Excessive use of the SCS will not only increase costs but also degrade system performance. Therefore, we use BRPG where the players tend to maximize/minimize a global objective function. As shown in (37), when the offloading ratio increases, the CRI of each edge is a differentiable convex function. Being the sum of (37), the trend of (39) must also be a convex distribution with increasing SCS utilization. When (39) reaches the global minimum, it is not only an optimal result but also the best time to terminate KPG algorithm. Therefore, as shown at the end of Function 2, we use Φ_{before} and Φ_{after} to check whether the utility of the overall system has reached the global minimum.

To prove that the offloading ratio analyzed by KPG is the optimal solution, we denote $R^* = \{\gamma_i^*,\ i \in I\}$ as the output of our proposed KPG and denote γ_i^* as the offloading ratio of the edge $i \in I$. We assume that the final global objective function is Φ^* and that an edge $j \in I$ can further optimize this result by increasing its offloading ratio γ_j. We do not discuss the impact of γ_j because we already proved that the global objective function must be strictly reduced during the iterations. Here, we prove R^* is the optimal solution by proving the irrationality of edge j.

Proof: Before the end of the iteration, we assume that the most cost-effective edge is k and that its value in (45) is s_k. The offloading ratio is r_k. When edge k increases its offloading ratio by a unit ($1/G$), the global objective function stops decreasing ($_{\triangle}\Phi > 0$) and the iteration generates the analyzed results (R^*). Please note that the final offloading ratio of edge k still remains at r_k, which is shown at the end of Function 2. Here, we consider an edge j, the value of which in (45) is s_j and its offloading ratio is r_j. Based on the above assumption, $s_j < s_k$, but edge j can cause the global objective function to undergo further optimization ($_{\triangle}\Phi < 0$). To compare edges

j and k, we denote their global objective functions as $_{\triangle}\Phi_j =_{\triangle} \Phi_{\bar{j}} +_{\triangle} \phi_j$ and $_{\triangle}\Phi_k =_{\triangle} \Phi_{\bar{k}} +_{\triangle} \phi_k$, where $_{\triangle}\Phi_{\bar{j}}$ and $_{\triangle}\Phi_{\bar{k}}$ are the sums of the objective functions of other edges impacted by edges j and k, respectively. Since s_j and s_k are fractions, they can be compared in terms of the following two cases: $\{|_{\triangle}\phi_j| = |_{\triangle}\phi_k|,\ _{\triangle}\gamma_j\lambda_{Lj} >_{\triangle} \gamma_k\lambda_{Lk}\}$ and $\{_{\triangle}\gamma_j\lambda_{Lj} =_{\triangle} \gamma_k\lambda_{Lk},\ |_{\triangle}\phi_j| < |_{\triangle}\phi_k|\}$. For the first case, because we know that $|_{\triangle}\phi_j| = |_{\triangle}\phi_k|$ and $_{\triangle}\Phi_k > 0$, the value of $_{\triangle}\Phi_{\bar{k}}$ must be larger than those of $_{\triangle}\phi_k$ and $_{\triangle}\phi_j$. However, since $_{\triangle}\gamma_j\lambda_{Lj} >_{\triangle} \gamma_k\lambda_{Lk}$, $_{\triangle}\Phi_{\bar{j}}$ must be larger than $_{\triangle}\Phi_{\bar{k}}$, which results in $_{\triangle}\Phi_j >_{\triangle} \Phi_k > 0$. For the other case, we know $_{\triangle}\gamma_j\lambda_{Lj} =_{\triangle} \gamma_k\lambda_{Lk}$; hence, $_{\triangle}\Phi_j$ must be larger than $_{\triangle}\Phi_k$ because $_{\triangle}\Phi_{\bar{j}} =_{\triangle} \Phi_{\bar{k}}$ but $|_{\triangle}\phi_{\bar{j}}| < |_{\triangle}\phi_{\bar{k}}|$. According to the results of the above two cases, we thereby prove that our solution R^* is the optimal solution with the irrationality of edge j.

4) Time complexity of the proposed KPG: To calculate the time complexity of our proposed KPG, we find the iterations in the functions and take the worst case into consideration. In Algorithm 1, a set of edges is initiated in a for loop, and the time complexity of this step is $O(N)$. Next, in the function of Bagging, a dual-layered iteration calculates an optimal offloading ratio set ($R^*_{n_0}$). For the outside iteration, when the capability of the SCS is extremely close to infinity, the worst-case of the time complexity appears because the best offloading ratios set is [1,1,...,1], as a result of which each edge should go forward $1 + 1/G$ steps. Thus, the worst case has $N \times [1 + (1/G)]$ iterations. On the other hand, for the inner iteration, searching for the most effective edge to increase the offloading ratio should take N steps, resulting in the worst-case number of iterations of the Bagging function being $N^2 \times [1 + (1/G)]$. Adding the steps of the initialization together with the Bagging function, the total number of iterations of our proposed KPG algorithm can be up to $N + N^2 \times [1 + (1/G)]$, and the time complexity of our proposed KPG is therefore $O(N^2)$. On the other hand, the time complexity of FKP is $Nlog(N)$. In the edge-cloud offloading mechanism, the time complexity of FKP is the same with KPG. Specifically, in each iteration, the cost-effective rate of each edge is changed as an edge adjusts its offloading ratio. FKP needs to find the most cost-effective edge again in each iteration. Thus, the time complexity of FKP in edge-cloud offloading scenario is $O(N^2)$ rather than $O(NlogN)$. Similarly, the time complexity of BRPG is $O(N^3)$; the maximum steps of the offloading ratio are $N[1 + (1/G)]$; and the time complexity of each step in BRPG is $O(N^2)$. Specifically, to guarantee the correctness of each step, BRPG calculates results for each edge as it increases the offloading ratio in units. In doing so, each edge takes $O(N)$, while N edges take $O(N^2)$. Thus, the overall iterations of BRPG are up to $N^2 \times N[1 + (1/G)]$, and the time complexity of BRPG is $O(N^3)$.

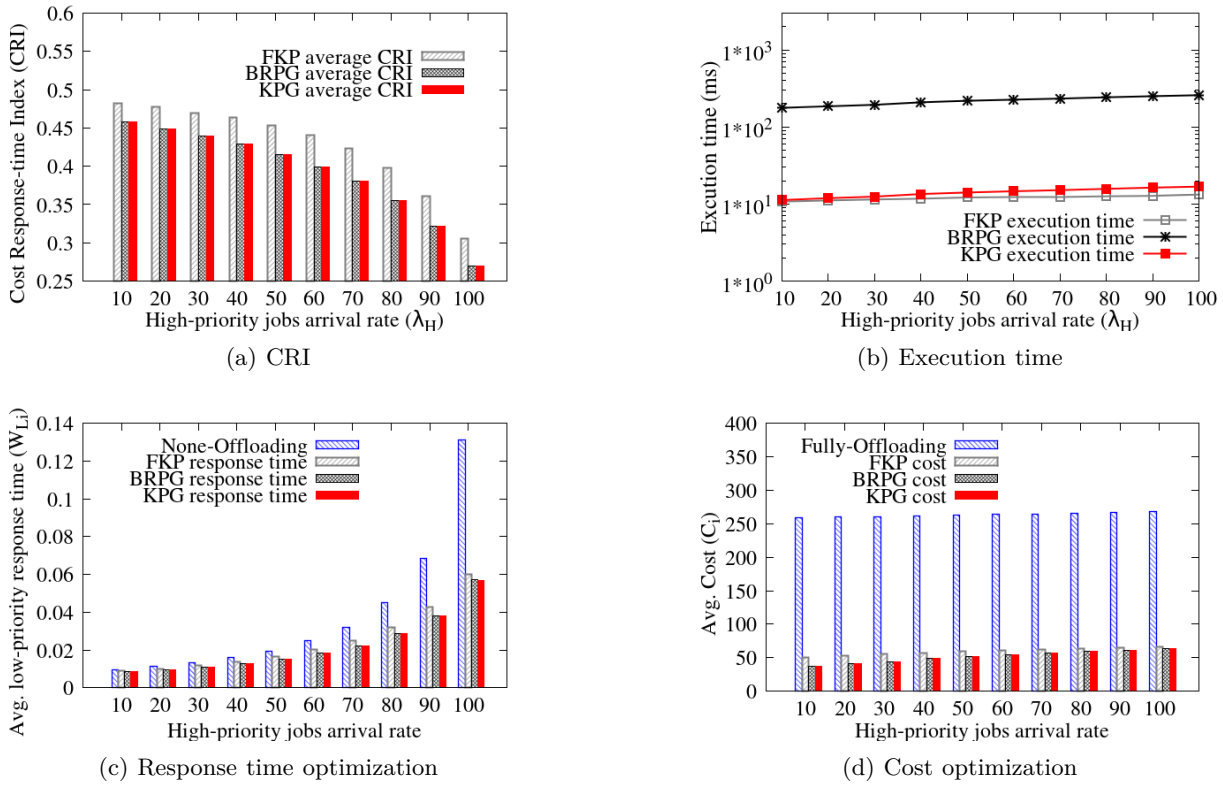
V. Performance Evaluation

In this section, we first evaluate the performance of KPG with two baseline algorithms:

1) FKP is typically solved by a greedy algorithm that can be formulated as an extensive form game [33] where players (edges) tend to put their chips (low-priority jobs) into the knapsack (SCS) selfishly. Compared with our proposed KPG algorithm, FKP does not pursue the global objective function but continue execution until no players want to change their decision. That is, the result of FKP is the pure Nash equilibrium, of which the process is fast but the performance is not always optimal.
2) BRPG is a cooperation-based algorithm, where players (edges) tend to maximize the utility of the overall system. BRPG checks the global objective function (44) based on the change in the strategy on each edge to obtain the optimal results. As discussed in Sec. IV-B3, although BRPG can find the optimal solution, checking the global objective function for each strategy increases the computational complexity significantly.

We conduct extensive simulations using ns-2 2.35 [34] with Ubuntu 18.04, 2 GB RAM, and i7 4770K CPU. According to AWS EC2 [35], the number of instances that a user can rent is up to 20 instances in each region. Since AWS divides its entire service region into 25 regions, the total number of instances that a user can use is up to 500. In addition, according to the rules of Microsoft Azure [36], each user can rent up to 800 instances. To evaluate the performance of KPG, we consider 10 MEC systems, each of which has the number of edges $N = 50$ and the number of SCS instances $n_0 = 4$. In addition, to test the stability and reliability of the proposed KPG, the arrival rates of low- and high-priority jobs $(\lambda_{Li}, \lambda_{Hi})$ are randomly assigned in the range $[10 : 200]$ for each MEC system. The service rates of the edge server and SCS follow an exponential distribution with the mean value $\mu_{Li} = \mu_{Hi} = \mu_c = 200$. In this way, we can observe that the behaviors of the system are from light to fully-loaded. Besides, according to the pricing policy of AWS EC2 [35], if the performance of an instance is two times better than that of another instance, the cost of the powerful instance will be two times than that of the general one. Thus, we consider the proportion of the costs between an edge and an SCS for about 1:4 ($c_{el} = c_{eh} = 0.1$, $c_b = 0.4$, $c_i = 0.2$) because the instances of an SCS is four and that of an edge is one in our experimental case.

(1) Impact of preemption: Our experimental results demonstrate that KPG has near-optimal performance in terms of CRI and execution time. CRI not only reflects the cost-effectiveness of the system but also represents the improvement compared with the worst case (CRI= 1). Therefore, in Fig. 6(a), we observe that the CRI decreases more obviously as the arrival rate of high-priority jobs increases. Specifically, as shown in Fig. 6(c) and Fig. 6(d), although both the response time and cost increase when the arrival rate of high-priority jobs increases, the improvement compared to the worst case is more significant. The worst cases discussed in Fig. 6(c)



(a) CRI (b) Execution time

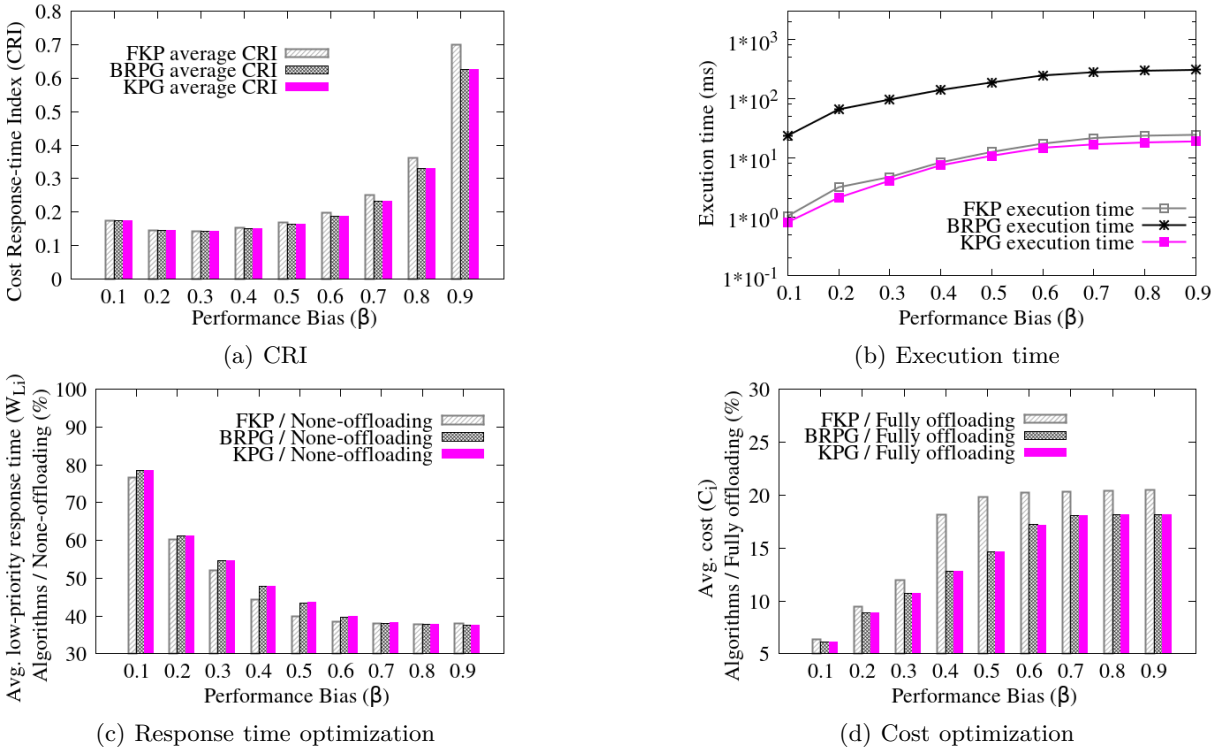
(c) Response time optimization (d) Cost optimization

Fig. 6: The impact of preemption: comparison of CRI, execution time, response time, and the cost of FKP, BRPG, and KPG in the worst case.

and Fig. 6(d) are None-Offloading and Fully-Offloading, respectively. For the response time, FKP, BRPG and KPG can improve upon that of the None-Offloading case by approximately 22%, 27.9% and 27.9%, respectively. For the cost, FKP, BRPG and KPG can improve upon that of the Fully-Offloading case by approximately 77.4%, 80.3% and 80.3%, respectively. Compared with FKP and BRPG on CRI, as shown in Fig. 6(a), our proposed KPG improves upon the result of FKP by approximately 8.5% and achieves 99.9% similarity with BRPG. On the other hand, for the execution time which is shown in Fig. 6(b), our proposed KPG improves upon the result of BRPG by approximately 93.5% and achieves similarity with FKP of approximately 83%.

(2) Impact of the performance bias: We are interested in the impact of the performance bias on the system. Fig. 7(a) and Fig. 7(b) depict the impact on CRI and the execution time, respectively. The average CRI has a convex distribution, which demonstrates that we can notify operators when their strategies are too extreme. Nevertheless, we do not intervene in the determination of the performance bias. We regard $\beta_i, i \in I$ as the input to our mathematical model and analyze the optimal offloading ratios based on the operator requirements. As mentioned in (37), the objective function in this paper consists of the response time and cost which are shown in Fig. 7(c) and Fig. 7(d), respectively. Specifically, when the performance bias $\beta < 0.5$, operators tend to cut costs and reduce the use of SCS; hence, little difference in the response time and cost results are observed among the three algorithms. However, when performance bias $\beta \geq 0.5$, operators tend to enhance service quality and increase offloading to SCS, which decreases the response time but increases costs. The results show that our proposed KPG is 99.9% similar to BRPG in terms of the average CRI, and that of FKP will deviate from the optimal solution by approximately 6.8%. For BRPG, however, even though the optimal result is achieved, the execution time is longer than that of the proposed KPG by approximately 94.86%.

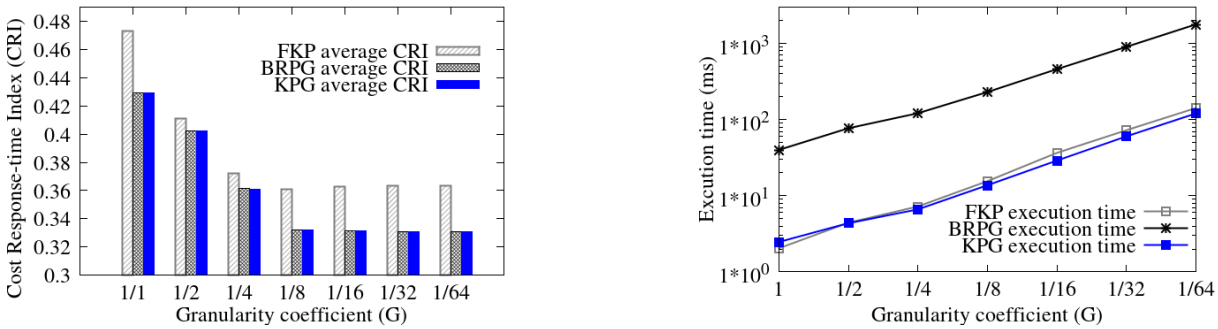
(3) Impact of the offloading precision: Our results show that KPG has near-optimal performance with respect to the average CRI and execution time regardless of the *Granularity* (G). Specifically, as shown in Fig. 8(a) and Fig. 8(b), when $G = 1$, the decision mechanism is binary, with entire jobs either completely solved at the edges or fully offloaded to SCS. When the *Granularity*(G) decreases, the results of the average CRI improve but the execution time increases slightly. However, even though an increase in precision can improve the performance, convergence eventually occurs. In this paper, we pay more attention to developing a flexible analytical model for operators. We do not intervene in the determination of the precision. Compared with the binary decision, with the unit of the offloading ratio in the $G = 1/8$ case ($min(\triangle\gamma) = 0.125$), the average CRI of KPG improves



(a) CRI (b) Execution time

(c) Response time optimization (d) Cost optimization

Fig. 7: The impact of performance bias: comparison of CRI, execution time, response time, and cost between FKP, BRPG, and KPG.



(a) CRI (b) Execution time

Fig. 8: The impact of precision requirement: comparison of CRI and execution time between FKP, BRPG, and KPG.

by approximately 22.7%, and the execution time needed increases from 2.46 ms to 13.72 ms. On the other hand, comparing our proposed KPG with FKP and BRPG in terms of CRI, our proposed KPG can achieve 99.9% similarity with BRPG and improves upon FKP by approximately 7%. Conversely, our proposed KPG can improve the execution time for BRPG by approximately 93.9% and that for FKP by approximately 7.6%.

VI. Conclusions

In this paper, we first study the impacts of the newly defined QoS priority on the offloading strategies in 5G MEC systems. Based on our investigation, we note new challenges (preemption, competition, and precision) due to the new standards in 3GPP R16. To address these challenges, we model the system and its performance metrics through comprehensive analytical models. Based on the models, we design KPG algorithm to find the optimal offloading ratios to balance the trade-off between cost and system performance. We demonstrate through extensive simulations that KPG has low computational complexity and outperforms FKP and BRPG in terms of several performance metrics. The results show that the performance of KPG is close to the optimal solution and that KPG can help operators design their offloading strategies in 5G MEC without large-scale deployment.

As for the future work, we will mainly focus on three topics. Firstly, we plan to further study the resource competition problem for multi-leveled QoS flows. Secondly, we will explore more system structures (e.g., clustering,

decentralization, layering, etc.) to increase the efficiency of the offloading mechanism. Thirdly, we will introduce more optimal algorithm to enhance system performance in terms of the cost and efficiency.

Acknowledgments

The research of Yi Ren was supported in part by EPSRC EP/T022566/1, EP/T024593/1, and the Royal Society IEC\R3\213100. Jyh-Cheng Chen's work was supported in part by the National Science and Technology Council of Taiwan under grant numbers 111-2221-E-A49-093-MY3, 111-2218-E-A49-023, and 111-3114-E-A49-001. The authors are very grateful to the editor and all reviewers for their constructive comments and valuable suggestions that significantly improved the quality of this paper.

References

[1] 3GPP TS23.203 v16.2.0, "Policy and charging control architecture (Release 16)," Std., Dec. 2019.
[2] A. Osseiran, S. Parkvall, P. Persson, A. Zaidi, S. Magnusson, and K. Balachandran, "5G wireless access: an overview," Ericsson White Paper, 2020.
[3] ITU TR M.2150-0, "Detailed specifications of the terrestrial radio interfaces of International Mobile Telecommunications-2020 (IMT-2020)," Tech. Rep., Feb. 2021.
[4] 3GPP TS38.213 v16.1.0, "Radio Access Network; NR; Physical layer procedures for control," Std., Mar. 2020.
[5] 3GPP TS38.331 v16.0.0, "Radio Access Network; NR; Radio Resource Control (RRC) protocol specification," Std., Mar. 2020.
[6] P. Paymard, N. Mokari, and M. Orooji, "Task scheduling based on priority and resource allocation in multi-user multi-task mobile edge computing system," in Proc. IEEE PIMRC, pp. 1–7, 2019.
[7] L. Gao and M. Moh, "Joint computation offloading and prioritized scheduling in mobile edge computing," in Proc. 2018 Int'l Conf. High Performance Computing Simulation (HPCS), pp. 1000–1007, 2018.
[8] J. Wang, W. Wu, Z. Liao, R. Simon Sherratt, G.-J. Kim, O. Alfarraj, A. Alzubi, and A. Tolba, "A probability preferred priori offloading mechanism in mobile edge computing," IEEE Access, vol. 8, pp. 39 758–39 767, 2020.
[9] A. Shakarami, A. Shahidinejad, and M. Ghobaei-Arani, "A review on the computation offloading approaches in mobile edge computing: A game-theoretic perspective," Software: Practice and Experience, vol. 50, 2020.
[10] X. Yang, X. Yu, H. Huang, and H. Zhu, "Energy efficiency based joint computation offloading and resource allocation in multi-access MEC systems," IEEE Access, vol. 7, pp. 117 054–117 062, 2019.
[11] X. Long, J. Wu, Y. Wu, and L. Chen, "Task merging and scheduling for parallel deep learning applications in mobile edge computing," in Proc. 2019 20th Int'l Conf. Parallel and Distributed Computing, Applications and Technologies (PDCAT), pp. 58–62, 2019.
[12] IEEE Std 1003.1-2017, "Portable Operating System Interface (POSIX(R)) Base Specifications," no. 7, 2018.
[13] S. Jošilo and G. Dán, "A game theoretic analysis of selfish mobile computation offloading," in Proc. IEEE INFOCOM, pp. 1–9, 2017.
[14] S. Jošilo and G. Dán, "Wireless and computing resource allocation for selfish computation offloading in edge computing," in Proc. IEEE INFOCOM, pp. 2467–2475, 2019.
[15] S. Sundar and B. Liang, "Offloading dependent tasks with communication delay and deadline constraint," in Proc. IEEE INFOCOM, pp. 37–45, 2018.
[16] W. Xiao, X. Zhu, W. Bao, L. Liu, and J. Yao, "Cooperative data sharing for mobile cloudlets under heterogeneous environments," IEEE Trans. Netw. Service Manag., vol. 16, no. 2, pp. 430–444, 2019.
[17] Y. Li, X. Wang, X. Gan, H. Jin, L. Fu, and X. Wang, "Learning-aided computation offloading for trusted collaborative mobile edge computing," IEEE Trans. Mob. Comput., vol. 19, no. 12, pp. 2833–2849, 2019.
[18] H. Wu, L. Chen, C. Shen, W. Wen, and J. Xu, "Online geographical load balancing for energy-harvesting mobile edge computing," in Proc. IEEE ICC, pp. 1–6, 2018.
[19] M. Guo, L. Li, and Q. Guan, "Energy-efficient and delay-guaranteed workload allocation in IoT-edge-cloud computing systems," IEEE Access, vol. 7, pp. 78 685–78 697, 2019.
[20] T. Alfakih, M. M. Hassan, A. Gumaei, C. Savaglio, and G. Fortino, "Task offloading and resource allocation for mobile edge computing by deep reinforcement learning based on SARSA," IEEE Access, vol. 8, pp. 54 074–54 084, 2020.
[21] H. Sun, H. Yu, G. Fan, and L. Chen, "QoS-aware task placement with fault-tolerance in the edge-cloud," IEEE Access, vol. 8, pp. 77 987–78 003, 2020.
[22] C. Joo and N. B. Shroff, "A novel coupled queueing model to control traffic via QoS-aware collision pricing in cognitive radio networks," in Proc. IEEE INFOCOM, pp. 1–9, 2017.
[23] S. Kang, C. Joo, J. Lee, and N. B. Shroff, "Pricing for past channel state information in multi-channel cognitive radio networks," IEEE Trans. Mob. Comput., vol. 17, no. 4, pp. 859–870, 2018.
[24] Zeng, Liu, Wang, and Lan, "Non-cooperative spectrum access strategy based on impatient behavior of secondary users in cognitive radio networks," Electronics, vol. 8, no. 9, 2019.
[25] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," IEEE/ACM Trans. Netw., vol. 24, no. 5, pp. 2795–2808, 2016.
[26] M. Chen, B. Liang, and M. Dong, "Joint offloading and resource allocation for computation and communication in mobile cloud with computing access point," in Proc. IEEE INFOCOM, pp. 1–9, 2017.
[27] M. Chen, M. Dong, and B. Liang, "Resource sharing of a computing access point for multi-user mobile cloud offloading with delay constraints," IEEE Trans. Mob. Comput., vol. 17, no. 12, pp. 2868–2881, 2018.
[28] M. Bouet and V. Conan, "Mobile edge computing resources optimization: A geo-clustering approach," IEEE Trans. Netw. Service Manag., vol. 15, no. 2, pp. 787–796, 2018.
[29] L. Huang, S. Bi, and Y. J. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," IEEE Trans. Mob. Comput., vol. 19, no. 11, pp. 2581–2593, 2019.
[30] A. Gandhi, M. Harchol-Balter, and I. Adan, "Server farms with setup costs," Perform. Eval., vol. 67, no. 11, pp. 1123–1138, 2010.
[31] M. H. Ishii Hiroaki, Ibaraki Toshihide, "Fractional knapsack problems," Math. Program., pp. 255–271, 1977.
[32] M. Voorneveld, "Best-response potential games," Econ. Lett., vol. 66, no. 3, pp. 289–295, 2000.
[33] H. Peters, "Extensive form games," Game Theory: A Multi-Leveled Approach, pp. 197–212, 2008.
[34] "The network simulator - ns-2." [Online]. Available: http://www.isi.edu/nsnam/ns/.
[35] "Cluster configuration guidelines and best practices" Available: https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-plan-instances-guidelines.html.
[36] "Resources not limited to 800 instances per resource group" Available: https://docs.microsoft.com/en-us/azure/azure-resource-manager/management/resources-without-resource-group-limit.

Cheng-Ying Hsieh received his B.S. degree in mechanical engineering from National Chiao Tung University (NCTU), Hsinchu, Taiwan, in 2014. He is now a Ph.D. candidate in the Department of Computer Science at National Yang Ming Chiao Tung University (NYCU), formerly NCTU. His research interests include 5G mobility networks, queueing analysis, deep learning.

Yi Ren received a Ph.D. degree in information communication and technology from the University of Agder, Norway, in 2012. He was with the Department of Computer Science, National Chiao Tung University (NCTU), Hsinchu, Taiwan, as a Postdoctoral Fellow, and an Assistant Research Fellow from 2012 to 2017. He is currently a Lecturer in the School of Computing Science at the University of East Anglia (UEA), Norwich, U.K. His research interests include IoT, imaging processing, AI, and mobile computing. His work involves both designing practical algorithms with sound theoretical foundations and building system prototype systems. He received the Best Paper Award at IEEE MDM 2012.

Jyh-Cheng Chen (S'96-M'99-SM'04-F'12) received the Ph.D. degree from the State University of New York at Buffalo in 1998. He was a Research Scientist with Bellcore/Telcordia Technologies, Morristown, NJ, USA, from 1998 to 2001, and a Senior Scientist with Telcordia Technologies, Piscataway, NJ, USA, from 2008 to 2010. He was with the Department of Computer Science, National Tsing Hua University (NTHU), Hsinchu, Taiwan, as an Assistant Professor, an Associate Professor, and a Full Professor from 2001 to 2008. He has been a Faculty Member with National Yang Ming Chiao Tung University (NYCU), formerly National Chiao Tung University (NCTU), since 2010. He is currently the Chair Professor with the Department of Computer Science, and the Dean of the College of Computer Science, NYCU. Dr. Chen is a Distinguished Member of the Association for Computing Machinery (ACM). He was a member of the Fellows Evaluation Committee, IEEE Computer Society. He has received numerous awards, including the Outstanding Teaching Award from both NCTU and NTHU, the Outstanding I.T. Elite Award, Taiwan, the Mentor of Merit Award from NCTU, the Medal of Honor and the K. T. Li Breakthrough Award from the Institute of Information and Computing Machinery, the Outstanding Engineering Professor Award from the Chinese Institute of Engineers, the Outstanding Research Award from the Ministry of Science and Technology, the Best Paper Award for Young Scholars from the IEEE Communications Society Taipei and Tainan Chapters, and the IEEE Information Theory Society Taipei Chapter, and the Telcordia CEO Award.