Multi-Service Edge Computing Management with Multi-Stage Coalition Game Task Offloading

Chun-Che Lin, *Student Member, IEEE*, Yao Chiang, *Member, IEEE*, and Hung-Yu Wei, *Senior Member, IEEE*

***Abstract*—The advent of 5G-enabled edge servers presents an opportunity to distribute computational tasks to the network edge. This approach helps alleviate the strain on limited central network resources caused by the rapid growth in the number of mobile devices and computation-intensive services. Moreover, it leads to reduced end-to-end delays for users. In this paper, we investigate resource allocation optimization in a dynamic multi-service system, where each service provider (SP) serves geographically dispersed service subscribers (SSs). Each SP can offload tasks to multiple edge servers, while each SS can freely switch between SPs offering homogeneous services. We propose the Multi-Stage Coalition Game Task Offloading (MSCGTO) framework, accommodating scalability, resource heterogeneity, and dynamic conditions. This framework encompasses two distributed algorithms to jointly maximize SP profit and minimize SS end-to-end delay, addressing cost-benefit considerations and user latency acceptance. We conduct extensive simulations and practical experiments with real-world services including augmented reality (AR), online gaming, and live video streaming applications, performed in a controlled testbed environment. The results of our experiments demonstrate that the proposed algorithms yield a 25% increase in system utility considering both the profit of SPs and the end-to-end delay of SSs when compared to existing approaches.**

***Index Terms*—Task offloading, edge computing, resource allocation, game theory, coalition game.**

I. Introduction

WITH the proliferation of mobile devices and the rapid expansion of extended reality (XR) services such as augmented reality (AR), virtual reality (VR), and mixed reality (MR), the global mobile data traffic has been experiencing exponential growth. According to Ericsson's 2022 report [1], the monthly global mobile network traffic surged from 55 exabytes (EB) to 108 EB between 2020 and 2022, doubling over a span of two years. Projections indicate that this trend will continue, with mobile data traffic anticipated to reach 453 EB by 2028, quadrupling over the course of six years. Video traffic is expected to dominate, accounting for 80% of the total mobile data traffic by 2028, representing a 10% increase from 2022. Moreover, XR services are expected to contribute significantly to this data load. Concurrently, the adoption of 5G technology is poised for substantial growth, with its share of total mobile subscriptions projected to increase from 17% in 2022 to 69% by 2028. The escalating mobile data traffic poses immense pressure on existing mobile networks, resulting in elevated end-to-end delays for service subscribers (SSs) and impeding service providers (SPs) in meeting quality of service (QoS) requirements. In response to these challenges, multi-access edge computing (MEC) [2] has emerged as a practical solution. By offloading computation to the network edge, MEC mitigates the strain on limited central network resources and reduces end-to-end delays for services. The lightweight nature of edge computing facilitates efficient distribution and management of computation and services across a diverse range of edge nodes, providing flexibility and scalability [3].

While MEC enables the delivery of services with greatly improved quality of service (QoS) [4] [5] [6] [7], the allocation of resources remains a challenging task due to the costs associated with edge servers and their limited computational capacity. Moreover, prevalent approaches often rely on centralized cloud servers for MEC resource admission control, raising concerns regarding privacy, security risks, a single point of failure, and substantial communication overhead [8] [9]. Distributed collaborative solutions have been proposed to tackle these issues. One such approach involves exploring collaborative caching between edge servers to enhance the cache hit ratio [10] [11] [12], thereby enabling more efficient utilization of the limited storage available in distributed edge nodes. Another aspect that has been discussed is the sharing of edge server resources among homogeneous SPs [13] [14] [15] [16]. These discussions employ game theoretical methods to model the interactions between SPs as a coalition game and seek to identify a Nash-stable partition. By sharing edge servers with each other, SPs can reduce infrastructure costs, optimize the utilization of underused edge servers, and ultimately increase their profits. However, these studies overlook the possibility of an SP serving geographically dispersed users and assume that each SP only utilizes a single edge server. Consequently, meeting the QoS requirements of distantly located SSs remains a challenge, thereby limiting the potential for SPs to enhance their profits. Furthermore, no existing study simultaneously considers the profitability of SPs and the QoS experienced by users in a multi-service system, while also evaluating the results using real-world data in a practical environment.

In light of these considerations, it is imperative to explore a profit-aware and collaborative approach within a more realistic system. Several challenges need to be addressed to achieve a viable solution. Firstly, SPs must have incentives to engage in collaborative efforts instead of individually renting and offloading tasks to edge servers. Secondly, when an SP shares its computation resources with another, the additional latency introduced by collaboration, or specifically, the increase in

The authors are grateful for the funding support from National Science and Technology Council (NSTC) of Taiwan under grant 112-2628-E-002-025-. The associate editor coordinating the review of this article and approving it for publication was H. Lutfiyya. (Corresponding author: Hung-Yu Wei.)

Chun-Che Lin, Yao Chiang, and Hung-Yu Wei are with the Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan (e-mail: b08901064@ntu.edu.tw; d05921015@ntu.edu.tw; hywei@ntu.edu.tw).

queuing delay due to serving more requests on the same set of edge server virtual machines (VMs), needs to be taken into account. Thirdly, if each SP is limited to offloading tasks to only one edge server, it will still be unable to meet the delay requirements of distant SSs. Fourthly, the system may encompass multiple services, making it challenging to manage a heterogeneous environment involving various SPs, SSs, and requests of diverse sizes and rates. Fifthly, in dynamic conditions where the switching cost for SSs is negligible, the system must consider the mobility of SSs, allowing them to freely switch between different SPs offering similar services to improve user acceptance. Sixthly, while collaborative approaches to resource allocation have received considerable attention, most existing works rely solely on simulation-based experiments. Therefore, it is essential to assess prospective solutions within a real-world setting, utilizing real-world data, to bridge the gap between theoretical progressions and practical implementation. Finally, the approach must exhibit high scalability to ensure that a real-world implementation, constrained by hardware limitations, can maintain acceptable latency even in large-scale scenarios.

Motivated by the aforementioned work and the challenges identified, we propose a Multi-Stage Coalition Game Task Offloading (MSCGTO) framework. The objective of this framework is to address the resource allocation optimization problem in a multi-service system that involves numerous edge servers, SPs, and SSs. In this system, SPs have the capability to offload their tasks to multiple edge servers through collaborative efforts. Initially, each SP serves a set of geographically distributed SSs, but later, the SSs are permitted to switch between SPs offering the same service. To tackle the complexity of this dynamic multi-hierarchical problem, we develop a two-stage coalition formation model. In the first stage, SPs providing the same service form and join multiple coalitions to share resources across multiple edge servers. The goal of these coalitions is to satisfy the demands of their respective SSs while maximizing their individual profits. This stage focuses on resource sharing and collaboration among SPs. In the second stage, the SSs evaluate the performance of the SPs offering the same service and make decisions to switch to the SP that offers the lowest delay. This stage aims to optimize the end-to-end delay experienced by the SSs by allowing them to select the most favorable SP based on their individual requirements. By decoupling the complex multi-hierarchical problems into these two stages, the proposed MSCGTO framework provides a systematic approach to address resource allocation challenges in a multi-service system of heterogeneous resources that aligns with cost-benefit considerations. It takes into account both the collaboration among SPs and the switching decisions of SSs to optimize the overall system performance and enhance the user experience and acceptance in terms of delay.

The contributions of this paper are summarized as follows:

- We propose a collaborative task offloading framework in a multi-service system, allowing each SP to leverage multiple edge servers and capitalize on the lightweight nature of edge deployment. This framework enables SPs to collaborate with other homogeneous SPs to serve geographically distributed SSs. Additionally, each SS is given the flexibility to switch between SPs. By adopting this mechanism, the profit of SPs can be maximized, and the delay experienced by SSs can be minimized.
- We formulate the resource allocation optimization problem as a two-stage coalition formation game and analyze it using game-theoretical principles. We develop two decentralized resource allocation algorithms to find stable coalition formations, which facilitate efficient collaboration among SPs and enhance the overall system performance.
- We conduct extensive simulation experiments to evaluate the performance of the proposed algorithms under various system configurations, revealing the MSCGTO framework to reach a 25% increase in system utility compared to existing approaches [16] [14]. Moreover, we substantiate the efficacy of our approach by conducting practical experiments using real services on a dedicated testbed.

The rest of this paper is organized as follows. Section II discusses related works. Section III describes the system model. Section IV presents the coalition formation models. Section V formulates the problem and presents two distributed algorithms to solve it. Section VI evaluates the performance of the proposed algorithms through comprehensive experiments. Section VII concludes the paper.

II. Related Works

In this section, we review existing literature on task offloading, service caching, and coalition formation approaches in resource allocation. We then compare these approaches to our proposed framework, highlighting its unique features and advantages.

A. Task Offloading and Service Caching

Task offloading with MEC has received significant research attention in recent years. Several studies have proposed innovative approaches to address different aspects of task offloading optimization. For instance, in [23], the authors introduce an efficient task offloading scheme based on software-defined networks, focusing on minimizing delay and conserving battery life. Similarly, [24] presents a comprehensive model for multi-user multi-task computation offloading in MEC, taking into account resource allocation, compression, and security concerns. In the context of an MEC-enabled multi-cell wireless network, [25] investigates the joint problem of task offloading and resource allocation to maximize the benefits for users. Additionally, [26] employs deep Q-learning to dynamically reconfigure the network slicing scale for optimizing task offloading efficiency. Furthermore, [27] introduces a novel approach that combines a multileader multifollower Stackelberg game and reinforcement learning to determine data offloading and pricing strategies in a risk-aware prospect-theoretic scenario.

Service caching has gained considerable attention in recent years as a means to reduce service latency and enhance user experience by caching services at the network edge. Researchers have explored various techniques and algorithms

TABLE I: **Literature Review**

#	Objective	Player	Service	User set	Substrate network	ES coalition granularity	Coalition membership	Requests	Testbed
[17]	SS utility	SS	Multiple	N/A	Multiple edges	Single coalition	Single coalition only	Simulated	X
[18]	device utility	device	Single	N/A	Single edge	N/A	Single coalition only	Simulated	X
[19]	device profit	device	Single	N/A	Multiple edges	Single coalition	Single coalition only	Simulated	X
[20]	BS utility	BS	Multiple	N/A	Multiple edges	N/A	Single coalition only	Simulated	X
[21]	BS utility	BS	Multiple	N/A	Multiple edges	N/A	Single coalition only	Simulated	X
[22]	UE computation	UE	Single	N/A	Single edge	Multiple coalitions	Single coalition only	Simulated	X
[15]	SP profit	SP	Multiple	Dynamic	Multiple edges	Single coalition	Single coalition only	Real data	X
[13]	SP utility	SP	Single	Static	Multiple edges	Single coalition	Single coalition only	Simulated	X
[14]	SP utility	SP	Single	Static	Multiple edges	Single coalition	Single coalition only	Simulated	O
[16]	SP profit	SP	Single	Static	Multiple edges	Multiple coalitions	Single coalition only	Simulated	X
Proposed	SP & SS utility	SP & SS	Multiple	Dynamic	Multiple edges	Multiple coalitions	Unrestricted	Real data	O

to optimize edge caching strategies. For instance, in [28], deep reinforcement learning and federated learning techniques are leveraged to enhance edge caching performance. The authors in [4] discuss different caching techniques in 5G mobile networks and propose an edge caching scheme based on content-centric or information-centric networking. Additionally, [29] introduces a context-aware proactive caching algorithm that considers the context information of connected users to update cache content and observe cache hits. These studies contribute valuable insights into the optimization of service caching in edge environments, aiming to improve latency and overall user satisfaction.

While the previously mentioned studies emphasize the use of MEC to enhance user quality of service through task offloading or service caching, they do not explicitly take into account the opportunity for collaboration among heterogeneous entities.

B. Coalition Game-Based Resource Allocation

Game theory is indeed a valuable tool for studying the interactions among players, whether they are SPs, SSs, or edge servers, in wireless networking systems. It offers a profit-aware alternative to central coordination by capturing the strategic behavior of individual players. Considering that each player is rational and self-interested, game theory enables the design of mechanisms that guide the system toward an equilibrium with desirable properties through the introduction of rules and constraints. When compared to other techniques such as reinforcement learning, game theory excels in modeling strategic interaction, cooperative dynamics, equitable resource allocation, Pareto optimality, and Nash equilibrium while maintaining interpretability.

Collaboration among players is often modeled using coalition formation games. In such games, players form coalitions to collaborate and share resources, such as spectrum sharing, collaborative service caching, or MEC resource sharing for joint task offloading. Coalition game solutions exhibit desirable properties, including individual rationality and Nash stability, implying that no player has an incentive to unilaterally deviate from the final solution. By employing coalition formation games, we can effectively model and analyze the collaborative dynamics in wireless networking systems, ensuring that the resulting solutions are stable and beneficial for all participants.

In the existing literature, there are several studies that employ coalition game theory to address resource allocation and collaboration in various wireless networking scenarios. In [17], a joint match-coalitional game-based algorithm is proposed to optimize resource allocation and improve efficiency and profits by allowing base stations (BSs) to form coalitions and share wireless and security resources. Similarly, [18] presents a collaborative task execution scheme for devices, ensuring individual rationality and promoting efficient resource utilization. In the context of blockchain networks, [19] applies coalition game theory to computation resource allocation, considering both individual and coalition profits. In the study presented in [20], the authors focus on service placement in MEC-enabled dense small cell networks using coalition formation game. By allowing BSs to collaborate and optimize service placement decisions, the operation cost of the edge system can be effectively reduced. In [21], a two-level stochastic hierarchical game is proposed to model the behavior of selfish BSs in networks where BSs are owned by different SPs. At the upper level, BSs participate in a coalition formation game to reach a stable partition, while at the lower level, stochastic subgames within each coalition are solved using reinforcement learning techniques. The authors of [22] exploit multi-carrier nonorthogonal multiple access (NOMA) enabled MEC systems using coalition formation game. They formulate user equipment (UE) as players and subcarriers as coalitions, achieving a Nash-stable solution through their proposed coalition game-based algorithm. In the context of cooperative service caching, [15] explores a hybrid service provisioning framework and utilizes hedonic game theory to design a dynamic coalition algorithm that enhances overall profit. Similarly, [13] demonstrates the significant reduction of social costs for SPs through cooperative service caching. In their scenario, SPs sharing the same edge server form a coalition, enabling them to share computation resources and costs. The practical effectiveness of distributed coalition game-based approaches is further investigated in [14] through experiments conducted on a real testbed. Lastly, in our previous work [16], we discuss a scenario where SPs have the capability to form multiple coalitions within a single edge server, thereby

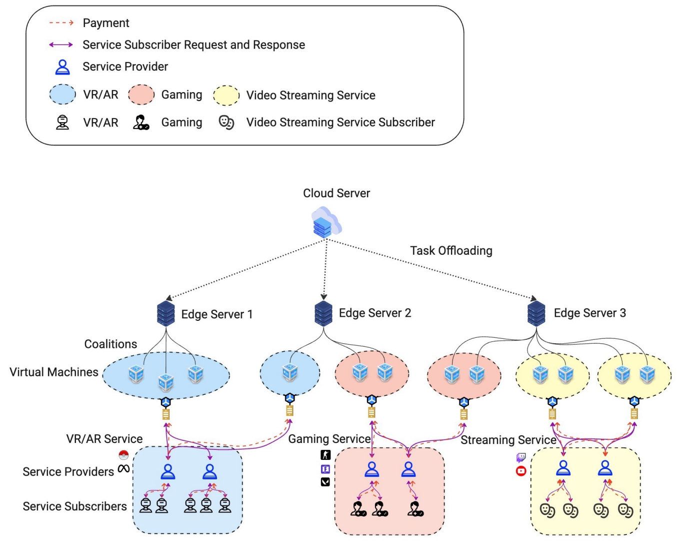


Fig. 1: The proposed multi-service coalition game-based task offloading system.

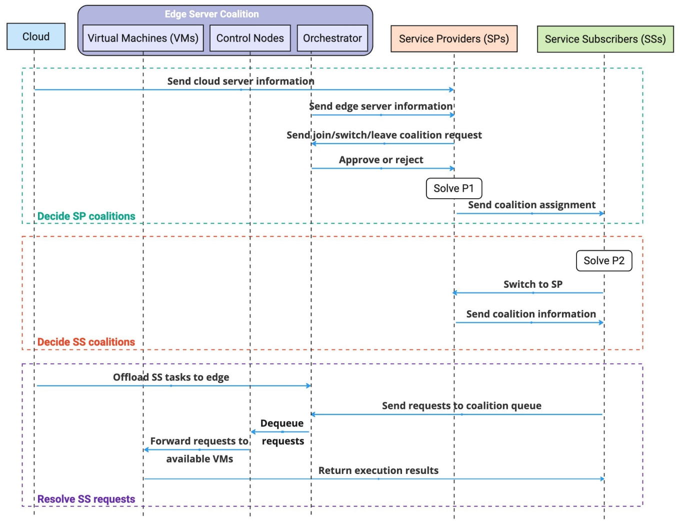


Fig. 2: The system workflow of the novel MSCGTO framework.

maximizing the utilization of edge server resources.

A comparison of related works using coalition game for edge server resource allocation optimization is presented in Table I. It is apparent that existing works have overlooked the potential for players to join multiple coalitions. Furthermore, none of the studies have considered the scenario where users have the flexibility to dynamically switch between different SPs to improve user acceptance, multiple coalitions can coexist within a single edge server to enhance scalability, and real-world data is utilized to evaluate the practicality of the proposed algorithms in a real-world environment.

III. System Model

In this section, we first provide an overview of the system architecture of the proposed MSCGTO framework. We then present the mathematical models that capture the important aspects of the system, including the delay of service, the revenue, cost, and profit of SPs, and the utility of SSs.

The system model depicted in Fig. 1 represents a multi-service system composed of various heterogeneous components. Specifically, the system includes three distinct services: VR/AR service, online gaming service, and live video streaming service. Each of these services imposes strict demands, requiring either low latency or high computational capabilities. The system includes four key components: a cloud server, M edge servers, N SPs, and S SSs. The set of servers, denoted as $\mathcal{M} = \{0, 1, \cdots, M\}$, encompasses the cloud server (0) and M edge servers ($1 \sim M$) based on the 3-tiered IEEE 1935 edge architecture [30]. Each of the edge servers consists of an orchestrator, control nodes and VMs, which are compute nodes on physical hosts. The orchestrator acts as the broker between the resources within its edge server and the SPs. The set of SPs, denoted as $\mathcal{N} = \{1, \cdots, N\}$, includes N entities ($1 \sim N$) responsible for providing services. The set of SSs is represented as $\mathcal{S} = \{1, \cdots, S\}$ and consists of S subscribers ($1 \sim S$) who require access to the services.

In the system model, each SP and SS is associated with one of the three services (VR/AR, online gaming, or live video streaming). During the first stage of the game, SPs belonging to the same service are permitted to form coalitions with each other. In this stage, each SS is initially assigned to an SP of the corresponding service. However, in the second stage, SSs have the flexibility to switch to another SP within the same service. This two-stage process allows for collaboration among SPs and enables SSs to optimize their SP selection. The overall workflow of the system is visualized in Fig. 2.

The edge servers in the system are offered by infrastructure providers, and SPs have the option to lease resources from them. When SPs leasing VMs from the same edge server decide to collaborate, they form an SP coalition. Within a coalition, SPs share their resources and jointly bear the associated costs. An SP can choose to directly obtain its service from the cloud server or offload its service to one or more edge servers to minimize latency. If an SP decides to utilize an edge server, it has the flexibility to decide whether to lease and pay for the VMs provided by the infrastructure providers or share the pool of VMs with other SPs that are also utilizing the same edge server. This sharing mechanism reduces costs and optimizes the utilization of idle resources. To cater to geographically distributed SSs and evenly distribute the load of SS requests, an SP can participate in multiple coalitions across different edge servers.

A. SP Delay Model

The delay model for the service of each SS is composed of two components: propagation delay and response delay. When an SS is served by the cloud, the response delay remains constant. However, if the SS is served by a coalition on an edge server, the response delay is determined by the total response time of an M/M/c queue. The delay is represented by the formula:

$$D_s = D_p^s + D_r^k \tag{1}$$

where

$$D_p^s = \alpha d_s \tag{2}$$

$$D_r^k = \begin{cases} \dfrac{E(\nu_k, \frac{\nu_k \cdot \mu}{\upsilon_k \cdot \lambda_k^{\mathcal{K}}})}{\nu_k \cdot \mu - \upsilon_k \cdot \lambda_k^{\mathcal{K}}} + \dfrac{1}{\mu}, & \text{if } 1 \le m \le M \\ \gamma, & \text{if } m = 0 \end{cases} \tag{3}$$

where

$$E(a,b) = \frac{1}{1 + (1-b)(\frac{a!}{b^a}) \sum_{i=0}^{a-1} \frac{a^i}{i!}} \tag{4}$$

where D_s is the total delay of the service of SS s received, k is the coalition SS s is in, D_p^s is the propagation delay from the server of SS s to itself, D_r^k is the total response delay of coalition k, α is the ratio of propagation delay to distance traveled, d_s is the distance between SS s and its server, m is the server coalition k is in, E is Erlang's C formula [31], which calculates the probability of a job in an M/M/c queue being sent into the queue rather than being served immediately, γ represents the response delay of the cloud server, ν_k is the number of VMs in coalition k, υ_k is the total number of SSs served by coalition k, μ is the processing rate of each VM, and $\lambda_k^{\mathcal{K}}$ is the request rate of each SS of the service offered by coalition k.

B. SP Revenue Model

The revenue of each SP is derived from the payments made by its SSs. An SS will pay its SP if the delay of the received service meets the agreed delay guarantee. Conversely, if the delay requirement is not met, the SP will have to pay a penalty to the SS.

The revenue of SP n can be formulated as:

$$R_n = \sum_{s \in \mathcal{S}_n} q_s \tag{5}$$

where

$$q_s = \begin{cases} p_s, & \text{if } x_s = 1 \\ -\beta p_s, & \text{if } x_s = 0 \end{cases} \tag{6}$$

where

$$x_s = \begin{cases} 1, & \text{if } D_s \le T_s \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

where R_n is the revenue of SP n, $\mathcal{S}_n$ is the set of SSs served by SP n, q_s denotes the payment made by SS s to its SP, p_s represents the service subscription price of SS s, $0 < \beta \le 1$ is the penalty-to-price ratio, x_s is a binary variable indicating whether the delay of the service received by SS s satisfies the agreed delay requirement or not, D_s is the delay of the service of SS s as defined in Eq. (1), and T_s is the delay requirement of SS s.

C. SP Cost Model

The cost incurred by each SP is associated with the payment made to infrastructure providers for leasing edge server VMs. The cost model is expressed as follows:

$$C_n = \sum_{k \in \mathcal{K}_n^{\mathcal{N}}} C_n^k \tag{8}$$

$$C_n^k = \begin{cases} \nu_k c^v \frac{\upsilon_k^n}{\upsilon_k}, & \text{if } 1 \le m \le M \\ 0, & \text{if } m = 0 \end{cases} \tag{9}$$

$$\upsilon_k = \sum_{n \in k} \upsilon_k^n \tag{10}$$

where C_n represents the total cost incurred by SP n to serve its SSs, C_n^k denotes the cost for SP n to use the resource of coalition k, $\mathcal{K}_n^{\mathcal{N}}$ is the set of coalitions in which SP n participates, ν_k denotes the number of VMs in coalition k, c^v signifies the cost associated with leasing a VM, υ_k is the total number of SSs served by coalition k, υ_k^n denotes the number of SSs served by SP n within coalition k, and m refers to the server in which coalition k is located. If the coalition exists on the cloud server (i.e., $m = 0$), the cost is zero since there is no expense incurred for utilizing edge server resources.

D. SP Profit Model

The profit of each SP is determined by the difference between its revenue and cost. The profit model can be expressed as:

$$P_n = R_n - C_n \tag{11}$$

where P_n represents the profit of SP n, R_n denotes the revenue of SP n as defined in Eq. (5), and C_n represents the cost incurred by SP n as defined in Eq. (8).

E. SS Utility Model

The utility of an SS is determined based on the normalized surplus of the delay of service received. It can be expressed as:

$$V_s = \frac{T_s - D_s}{T_s} p_s x_s \tag{12}$$

where x_s is the binary variable defined in Eq. (7). The surplus of the delay of service, which is the difference between the delay requirement T_s and the actual delay of the service received D_s, is normalized with respect to T_s and multiplied by the subscription price p_s. This formulation allows for the optimization of utility along with the profit of the SPs, as defined in Eq. (11). It is important to note that the utility is zero if the delay requirement is not met.

IV. Coalition formation models for SPs and SSs

In this section, we delve into the coalition formation models tailored for the proposed MSCGTO framework. These models play a crucial role in optimizing resource allocation, fostering collaboration among SPs, and ensuring the satisfaction of SSs. Building upon the profit considerations for SPs and the utility of SSs discussed in Section III, we present the comprehensive models that encapsulate the dynamic aspects of the multi-service environment through coalitional game theory.

A. SP Collaborative Task Offloading Model

The collaboration between SPs is investigated through a coalition game, which is formulated as follows:

- Player: There are N SPs, and they are the players.
- Strategy: Each SP has 3 strategies.
  1) Obtain services from the cloud server.
  2) Offload SS requests by leasing VMs on an edge server.
  3) Offload SS requests by joining one or more coalitions on edge servers to share VMs with other coalition members.
- Utility: The utility of an SP is determined by its profit, as defined in Eq. (11).

An SP coalition is a group of SPs sharing the same set of VMs and the associated costs on an edge server. It can consist of one or more SPs. These coalitions can vary in size, consisting of one or more SPs, and multiple coalitions can coexist within a single edge server. Moreover, each SP has the flexibility to join multiple coalitions across different edge servers, enabling them to enhance their profits.

To formally define the coalition games within this system, we provide the following definitions:

Definition 1. *A coalition formation game for SPs can be defined as a Triplet $(\mathcal{N}, \mathcal{C}, U^{\mathcal{N}})$, where $\mathcal{N}$ is the set of SPs, $\mathcal{C}$ is a collection of SP coalitions, and $U^{\mathcal{N}}$ denotes the coalition utility of SPs.*

Definition 2. *A collection of SP coalitions $\mathcal{C}$ is defined as any arbitrary set of coalitions $K_i^{\mathcal{N}} \subset \mathcal{N}$, i.e., $\mathcal{C} = \{K_1^{\mathcal{N}}, K_2^{\mathcal{N}}, ..., K_i^{\mathcal{N}}, ..., K_K^{\mathcal{N}}\}$, such that coalitions can overlap. If $\mathcal{C}$ covers the entire SP set $\mathcal{N}$, i.e., $\bigcup_{i=1}^{K} K_i^{\mathcal{N}} = \mathcal{N}$, $\mathcal{C}$ can also be regarded as a covering of $\mathcal{N}$.*

Definition 3. *$U_n^{\mathcal{N}}(\mathcal{E})$ is the utility of SP $n \in \mathcal{N}$ under the SS request distribution strategy set $\mathcal{E}$ for all SPs. $\mathcal{E}$ specifies how each SP distributes its SS requests among its multiple coalitions, ensuring that each request is served by one and only one coalition.*

An SP can only leave or join a coalition when it will not reduce other SPs' utilities. In other words, each operation needs to be a Pareto improvement to the system. If SPs are allowed to frequently leave or join in a way that hurt others, the long-term success of the system will be undermined. However, by only allowing changes that are Pareto improvements, each SP is guaranteed to at least maintain its current utility, and possibly increase it. This ensures that no SP will be worse off from collaboration, providing a strong incentive for SPs to participate in the coalition game.

Based on the coalition formation game model [32], we introduce the Pareto-based preference relation of SPs as below:

Definition 4. *(Pareto-Based Preference Relation for SPs) For any SP $n \in \mathcal{N}$, we propose the following preference relation:*

$$\begin{aligned} & \mathcal{K}_n^{\mathcal{N}'} \succ_n^{IS} \mathcal{K}_n^{\mathcal{N}} \iff \\ & U_n^{\mathcal{N}}(\mathcal{E}') > U_n^{\mathcal{N}}(\mathcal{E}) \\ & U_j^{\mathcal{N}}(\mathcal{E}') \geq U_j^{\mathcal{N}}(\mathcal{E}) \; \forall j \in K_i^{\mathcal{N}} \; \forall K_i^{\mathcal{N}} \in \mathcal{K}_n^{\mathcal{N}'} \end{aligned} \tag{13}$$

where $\mathcal{K}_n^{\mathcal{N}}$ denotes the set of coalitions SP n is originally in, $\mathcal{K}_n^{\mathcal{N}'}$ denotes the new set of coalitions of SP n, and $\mathcal{E}'$ is the SS request distribution strategy set after the set of coalitions SP n is in becomes $\mathcal{K}_n^{\mathcal{N}'}$. This means that an SP will prefer the coalition set $\mathcal{K}_n^{\mathcal{N}'}$ to $\mathcal{K}_n^{\mathcal{N}}$ if and only if its profit can be increased and if the profit of any other SP in the coalitions affected won't be jeopardized, including the SPs that share at least one coalition with SP n.

Definition 5. *(Join and Leave Rule for SPs) For any SP $n \in \mathcal{N}$, it will join the coalition $K_{i'}^{\mathcal{N}}$ if and only if $\{\mathcal{K}_n^{\mathcal{N}} \cup \{K_{i'}^{\mathcal{N}}\}\} \succ_n^{IS} \mathcal{K}_n^{\mathcal{N}}$, where $\mathcal{K}_n^{\mathcal{N}}$ denotes the set of coalitions SP n is originally in. When the preference relation is satisfied, joining $K_{i'}^{\mathcal{N}}$ will be a Pareto improvement of the system. Similarly, it will leave $K_{i'}^{\mathcal{N}}$ if and only if $\{\mathcal{K}_n^{\mathcal{N}} \backslash K_{i'}^{\mathcal{N}}\} \succ_n^{IS} \mathcal{K}_n^{\mathcal{N}}$. When the preference relation is satisfied, leaving $K_{i'}^{\mathcal{N}}$ will be a Pareto improvement of the system.*

B. SS Hedonic Coalition Formation Model

After the SPs have established a stable coalition structure for collaborative task offloading, the SSs have the flexibility to switch between SPs that offer the same type of service in order to maximize their own utility. While an SP can join multiple computation coalitions, an SS of the SP can only send its requests to one of the coalitions. Thus, when an SS decides to switch between different SPs, it is essentially choosing which computation coalition to join. In light of this, we can formulate the coalition game as follows:

- Player: There are S SSs, and they are the players.
- Strategy: Each SS can decide which SP to subscribe to, essentially deciding which underlying computation coalition to use.
- Utility: The utility of an SS is defined in Eq. (12).

The set of SSs using the same computation coalition, i.e., the same set of VMs, is defined as a coalition. In the following, we discuss some properties of the coalition game:

Property 1. *The proposed SS coalition game is a non-transferable utility (NTU) game.*

In the SS coalition game, the utility of each SS is determined by the delay of service received, which is specific to each individual SS. This means that the utility cannot be transferred between SSs. Therefore, the game satisfies the criteria of an NTU game [33], where the utility cannot be redistributed among the players.

Property 2. *The proposed SS coalition game is a class of hedonic game.*

A hedonic game is a type of NTU game in which players have preferences over which coalition they belong to, and their utility depends solely on the composition of the coalition they are in, rather than how other players are grouped. In the proposed SS coalition game, each SS has a preference over the computation coalition they join, and their utility is determined by the delay of service received within that coalition, independent of how other players are partitioned or the specific coalitions they form, since the computation load of different sets of VMs does not affect each other in any way. This aligns with the characteristics of a hedonic game [34]. Additionally, hedonic games typically involve self-interested players who make choices based on their individual preferences, which is applicable in this scenario as SSs aim to maximize their utility by switching between SPs. Therefore, the proposed SS coalition game satisfies the properties of being a hedonic game.

We define the SS coalition game as follows:

Definition 6. *A coalition formation game for SSs can be defined as a Triplet $(\mathcal{S}, \mathcal{P}, U^{\mathcal{S}})$, where $\mathcal{S}$ is the SS set, $\mathcal{P}$ is a collection of SS coalitions, and $U^{\mathcal{S}}$ denotes the coalition utility of SSs.*

Definition 7. *A collection of SS coalitions $\mathcal{P}$ is defined as any arbitrary set of disjoint coalitions $K_i^{\mathcal{S}} \subset \mathcal{S}$ i.e. $\mathcal{P} = \{K_1^{\mathcal{S}}, K_2^{\mathcal{S}}, ..., K_i^{\mathcal{S}}, ..., K_S^{\mathcal{S}}\}$, such that $\forall i \neq i'$, $K_i^{\mathcal{S}} \cap K_{i'}^{\mathcal{S}} = \emptyset$. If $\mathcal{P}$ spans the set of SSs $\mathcal{S}$, i.e., $\bigcup_{i=1}^{S} K_i^{\mathcal{S}} = \mathcal{S}$, $\mathcal{P}$ can also be regarded as a partition of $\mathcal{S}$.*

Definition 8. *A solo coalition $\{s\}$ contains only the unserved SS s. The set of solo coalitions $\tilde{S}_{\mathcal{K}}$ under set $\mathcal{K}^{\mathcal{S}}$ is defined as $\mathcal{S} \backslash \{\mathcal{S} \cap \mathcal{K}^{\mathcal{S}}\}$. $\tilde{S}_{\mathcal{K}}$ is a subset of $\mathcal{S}$.*

Definition 9. *$U_s^{\mathcal{S}}(K_s^{\mathcal{S}})$ is the utility of SS s in coalition $K_s^{\mathcal{S}}$.*

Definition 10. *(Preference Relation for SSs) For any SS $s \in K_i^{\mathcal{S}}$ where $i \in \mathcal{N}$, we propose the following preference relation:*

$$K_i^{\mathcal{S}} \succ_s K_j^{\mathcal{S}} \iff U_s^{\mathcal{S}}(K_i^{\mathcal{S}}) > U_s^{\mathcal{S}}(K_j^{\mathcal{S}}) \ \forall j \in \mathcal{N} \tag{14}$$

$$U_s^{\mathcal{S}}(K^{\mathcal{S}}) = \begin{cases} V_s, & \text{if } K^{\mathcal{S}} \notin \mathcal{H}(s) \\ -\infty, & \text{else} \end{cases} \tag{15}$$

where V_s is the utility of SS s defined in Eq. (12), and $\mathcal{H}(s)$ denotes the history set of the coalitions SS s has joined and left. This means that SS s will prefer coalition $K_i^{\mathcal{S}}$ to $K_j^{\mathcal{S}}$ if and only if its preference value can be improved by splitting from $K_i^{\mathcal{S}}$ as a solo coalition $\{s\}$ and merging with coalition $K_j^{\mathcal{S}}$. Since the preference value is $-\infty$ for coalitions in the history set, SS has no incentive to revisit a coalition. It can be seen as a learning process.

Definition 11. *(Split and Merge Rule for SSs) For any SS $s \in K_i^{\mathcal{S}}$, it will split from $K_i^{\mathcal{S}}$ and merge with $K_j^{\mathcal{S}}$, if and only if $\{K_j^{\mathcal{S}} \cup \{s\}\} \succ_s K_i^{\mathcal{S}}$, where $i, j \in \mathcal{N}$.*

TABLE II: **List of Notations**

Notation	Meaning
$\mathcal{N}, \mathcal{M}, \mathcal{S}$	The set of SPs, edge servers, and SSs.
N, M, S	The number of SPs, edge servers, and SSs.
$\mathcal{K}^{\mathcal{N}}$	The set of SP coalitions.
K	The number of SP coalitions.
$\mathcal{K}_n^{\mathcal{N}}$	The set of coalitions that SP n is in.
$K_k^{\mathcal{N}}$	The set of SPs using coalition k.
$K_n^{\mathcal{S}}$	The set of SSs using SP n.
$\sigma_i^{\mathcal{N}}, \sigma_i^{\mathcal{S}}$	The service type of SP i and SS i.
$\lambda_k^{\mathcal{K}}$	Request rate of each SS of the service of coalition k.
μ	Processing rate of each VM.
γ	Response delay of the cloud.
ν_k	The number of VMs in each coalition.
υ_k	The total number of SSs served by coalition k.
υ_k^n	The number of SSs of SP n served by coalition k.
α	Delay-distance ratio.
β	Price-penalty ratio.
c^v	Cost of a VM.
$\mathcal{A}$	The matrix of SP offloading decisions.
$\mathcal{B}$	The vector of SS subscribing decisions.
$\mathcal{E}$	The vector of SS coalition assignments.

V. Problem Formulation

In this section, we will mathematically formulate the objective of our work, which is to optimize the utility of the system. Given the complexity and multiple hierarchies present in the system, we later decompose the original problem into two distinct subproblems.

For clarity, the main notations involved in our system are summarized in Table II.

A. Overall Problem Formulation

The optimization goal is to maximize the total utility of the system under various constraints, which can be formulated as

$$\begin{aligned} &\max_{\mathcal{A},\mathcal{B}} \left(\sum_{n\in\mathcal{N}} (R_n - C_n) + \sum_{s\in\mathcal{S}} \frac{T_s - D_s}{T_s} p_s x_s \right) \\ &\text{s.t.} \\ &C1: a_{nk} \in \{0,1\} \\ &C2: \exists k \in \mathcal{K}_n^{\mathcal{N}} \ s.t. \ a_{nk} = 1 \ \forall n \in \mathcal{N} \\ &C3: \sigma_i^{\mathcal{N}} = \sigma_j^{\mathcal{N}} \ \forall i, j \in \mathcal{K}_n^{\mathcal{N}} \ \forall n \in \mathcal{N} \\ &C4: 1 \leq b_s \leq N \\ &C5: \sigma_i^{\mathcal{S}} = \sigma_j^{\mathcal{S}} \ \forall i, j \in K_n^{\mathcal{S}} \ \forall n \in \mathcal{N} \end{aligned} \tag{16}$$

where $\mathcal{A} = \{a_{nk} | n \in \mathcal{N}, k \in \mathcal{K}^{\mathcal{N}}\}$ and $\mathcal{B} = \{b_s | s \in \mathcal{S}\}$, expressing the strategies of SPs and SSs. $a_{nk} = 1$ if SP n participates in coalition k, and $a_{nk} = 0$ if not. b_s denotes the SP that serves SS s. $\sigma_i^{\mathcal{N}}$ and $\sigma_i^{\mathcal{S}}$ are the service type of SP i and SS i respectively, The first component is the sum of the profit of all SPs defined in (11), and the second component is the sum of the utility of all SSs defined in (12). Constraints C1 and C2 state that for every coalition, each SP is either inside the coalition or not, and that each SP needs to be in at least one coalition. Constraint C3 ensures that every SP inside a coalition belongs to the same type of service. Constraint C4 guarantees that each SS can only be served by one SP. Constraint C5 requires all SSs served by the same SP to be subscribing to the same type of service.

Due to the involvement of different hierarchies in the decision-making processes of $\mathcal{A}$ and $\mathcal{B}$, we will decompose the original problem into two subproblems. This decomposition allows us to address the challenges faced by SPs and SSs separately, solving them sequentially in a more efficient manner. We will formulate and optimize the subproblems individually to achieve our overall objective.

B. First Stage: Pareto-Base Coalition Formation Game for Service Providers

1) maximize total SP profit, which can be formulated as:

$$\textbf{P1:} \quad \begin{aligned} &\max_{\mathcal{A}} \sum_{n\in\mathcal{N}} (R_n - C_n) \\ &\text{s.t. } C1, C2, C3 \end{aligned} \tag{17}$$

SPs aim to maximize their individual profits by deciding which coalitions to join. This process leads to a stable coalition formation, where no SP has the incentive to leave its current coalitions or join another coalition, based on the Pareto-based preference criterion described in Definition 5.

To address this problem, we propose an efficient distributed algorithm for solving the coalition game and achieving an individually stable outcome. The algorithm begins by initializing the locations of the servers and the SSs of the SPs. Initially, all SPs provide their services from the cloud. Each SP then has the opportunity to offload its tasks to one or more edge servers, optimizing its profit by joining or leaving coalitions. The algorithm continues until no SP has any incentive to deviate from its chosen strategy. At this point, the equilibrium is reached, and the algorithm terminates. For a detailed step-by-step description of the algorithm, refer to Algorithm 1.

We can prove the convergence and stability properties as follows.

Definition 12. *A covering $\mathcal{C}$ is individually stable if*

$$\begin{aligned} &\mathcal{K}_n^{\mathcal{N}} \succ_n^{IS} \{\mathcal{K}_n^{\mathcal{N}} \backslash K_i^{\mathcal{N}}\} \\ &\forall K_i^{\mathcal{N}} \in \mathcal{K}_n^{\mathcal{N}} \ \forall n \in \mathcal{N} \end{aligned} \tag{18}$$

$$\begin{aligned} &\mathcal{K}_n^{\mathcal{N}} \succ_n^{IS} \{\mathcal{K}_n^{\mathcal{N}} \cup \{K_i^{\mathcal{N}}\}\} \\ &\forall K_i^{\mathcal{N}} \in \{\mathcal{C} - \mathcal{K}_n^{\mathcal{N}}\} \end{aligned} \tag{19}$$

When the aforementioned conditions are met, no SP has any incentive to leave its existing coalitions or join another coalition, as determined by the Pareto-based join and leave rule outlined in Definition 5.

Theorem 1. *Algorithm 1 is guaranteed to converge to a final coalition formation.*

Proof. Based on the preference criterion outlined in Definition 4, a join or leave operation is only performed if it results in a Pareto improvement, meaning that each switch leads to a new and unvisited covering with a higher system utility. Since there is a finite number of coalitions and SPs, the number of possible coverings is also finite. As a result, the algorithm is guaranteed to reach a final coalition formation. □

Theorem 2. *The final coalition formation of Algorithm 1 is individually stable.*

Proof. Suppose the final coalition formation achieved through Algorithm 1 is not individually stable, indicating the presence of an SP with an incentive to join or leave a coalition according to the Pareto-based rules described in Definition 5. However, if such an action is taken by the SP, it would result in a new coalition formation, which contradicts our initial assumption that the previous coalition formation was the final one. This contradiction implies that the final coalition formation obtained from Algorithm 1 is indeed individually stable. □

Algorithm 1 Coalition formation algorithm for collaborative task offloading

```
1:  Initialization:
2:      Set the positions of servers and SPs.
3:      Set the SSs for each SP.
4:  repeat
5:      Select an SP $n \in \mathcal{N}$ via a predetermined permutation
    and find its current set of coalitions $\mathcal{K}_n^{\mathcal{N}}$.
6:      Uniformly randomly choose a coalition $K_i^{\mathcal{N}} \in \mathcal{K}_n^{\mathcal{N}}$.
7:      if $\{\mathcal{K}_n^{\mathcal{N}} \backslash K_i^{\mathcal{N}}\} \succ_n^{\text{IS}} \mathcal{K}_n^{\mathcal{N}}$ then
8:          Make SP $n$ leave coalition $K_i^{\mathcal{N}}$.
9:      end if
10:     Uniformly randomly choose a server $m \in \mathcal{M}$.
11:     for all coalition $K_j^{\mathcal{N}}$ in server $m$ do
12:         if $\{\mathcal{K}_n^{\mathcal{N}} \cup \{K_j^{\mathcal{N}}\}\} \succ_n^{\text{IS}} \mathcal{K}_n^{\mathcal{N}}$ then
13:             Make SP $n$ join coalition $K_j^{\mathcal{N}}$.
14:         end if
15:     end for
16: until No SP has the incentive to change its strategy.
17: Output: The stable SP offloading decisions set $\mathcal{A}$.
```

The collaborative task offloading scheme poses a challenging problem known as mixed-integer nonlinear programming (MINLP), which falls into the NP-hard category. The time complexity per iteration in Algorithm 1 exhibits a worst-case scenario of $O(N^2M^2)$, where N stands for the number of SPs and M stands for the number of edge servers. Additionally, the upper bound of the number of iterations required to reach a solution is exponential.

C. Second Stage: Hedonic Coalition Formation Game for Service Subscribers

1) maximize total SS utility, which can be formulated as:

$$\textbf{P2:} \quad \begin{aligned} &\max_{\mathcal{B}} \sum_{s\in\mathcal{S}} \frac{T_s - D_s}{T_s} p_s x_s \\ &\text{s.t. } C4, C5 \end{aligned} \tag{20}$$

After SPs have determined their offloading strategies, SSs have the opportunity to choose their preferred SP, indirectly selecting the computation coalition to which they will send their requests. Their decision is based on selecting the SP that can offer the lowest service delay. This incentivizes SSs to switch to the SP that can provide the most efficient computation coalition for their needs. Consequently, they are ensured to reach a stable coalition formation where no SS has the motivation to switch its SP, according to the split and merge rule outlined in Definition 10.

To address the challenge from the standpoint of SSs, we present an efficient distributed algorithm designed to solve the coalition game and attain a Nash-stable outcome. The algorithm builds upon the final coalition formation determined by Algorithm 1 and utilizes it as the initial state. Each SS is given the chance to switch to an alternative SP that offers a lower delay, thereby improving its utility. The algorithm continues until no SS has any incentive to deviate from their selected strategy. This signifies the achievement of a Nash equilibrium, and the algorithm concludes. The algorithm is presented in Algorithm 2.

The convergence and stability properties can be established through the following proof.

Definition 13. *A partition $\mathcal{P}$ is Nash-stable if*

$$\begin{aligned} &K_i^{\mathcal{S}} \succ_s K_j^{\mathcal{S}} \cup \{s\} \\ &\forall s \in K_i^{\mathcal{S}} \ \forall K_i^{\mathcal{S}}, K_j^{\mathcal{S}} \in \mathcal{P}, i \neq j \end{aligned} \tag{21}$$

Under the condition described above, no SS has the incentive to switch to a different coalition, as dictated by the split and merge rule defined in Definition 11.

Theorem 3. *Algorithm 2 is guaranteed to converge to a final partition.*

Proof. Based on the preference relation defined in Definition 10, each switch operation results in a new and unvisited partition. It is important to note that the number of possible partitions of a set is finite, known as the Bell number. Hence, the algorithm is ensured to reach a final partition within a finite number of steps. □

Theorem 4. *The final partition of Algorithm 2 is Nash-stable.*

Proof. Let's assume that the final partition obtained from Algorithm 2 is not Nash-stable, implying that there is at least one SP with the incentive to switch its coalition. However, when the SP decides to switch, a new partition is formed, which contradicts our initial assumption that the previous partition was the final one. This contradiction indicates that the final partition achieved by the algorithm is indeed Nash-stable, where no SS has any incentive to switch its coalition under the defined split and merge rules in Definition 11. □

The utility maximization scheme for SSs is also a MINLP problem, which is NP-hard. In Algorithm 2, the worst-case time complexity per iteration is $O(N^2M)$, while the upper bound of the number of iterations is exponential.

Algorithm 2 Hedonic coalition formation algorithm for SS coalition assignments

```
1: repeat
2:     Select an SS $s \in \mathcal{S}$ via a predetermined permutation
       and find its current SP $n$.
3:     Uniformly randomly choose an SP $n' \in \mathcal{N}$.
4:     if $K_{n'}^{\mathcal{S}} \succ_s K_n^{\mathcal{S}}$ then
5:         Make SS $s$ switch to SP $n'$.
6:     end if
7: until No SSs has the incentive to change its strategy.
8: Output: The stable subscription decision set $\mathcal{B}$.
```

VI. Experimental Results

In this section, we first provide an overview of the experimental environment for evaluating the proposed scheme. We describe the system parameters, the implementation of the experimental testbed, and the utilization of real data. We then proceed to evaluate the scheme by comparing it against existing algorithms under different system configurations.

A. Experiment Setup

We consider a system consisting of 1 cloud server, M edge servers, N SPs, and S SSs. The system offers three types of services: VR/AR service, online gaming service, and live video streaming service. SPs and SSs are uniformly distributed to different services, and SSs of each service are uniformly distributed to SPs of the same service. The penalty-to-price ratio is 0.4. Each VM has a processing rate of 1.5GB/s, and the cost of renting a VM is \$20. Additionally, each edge server has the capacity to support 7 VMs at this processing rate.

Based on empirical studies conducted on VR/AR service [35], online gaming service [36], and live video streaming service [37], we have determined the following service-dependent parameters. The subscription prices are \$50, \$25, and \$20 for VR/AR, online gaming, and live video streaming respectively. The delay requirements for SSs are 20ms, 100ms, and 1s. The request throughputs for each SS are 100MB/s, 6MB/s, and 1MB/s. Finally, the request rates for each SS are 120 requests/s, 60 requests/s, and 30 requests/s respectively. These parameters have been thoughtfully selected based on empirical research and will be utilized in evaluating the performance of the proposed scheme during the subsequent experiments. The simulation has been implemented in Python 3.10 and is executed within an Ubuntu 22.04 LTS environment, utilizing an Intel i7-10700k CPU. We performed each experiment twenty times and calculated the means and standard errors to ensure the consistency of the outcomes. The error bars in the figures represent the 95% confidence interval of the results.

To assess the practical effectiveness of the proposed MSCGTO scheme, we conducted performance evaluations using real-life applications in our dedicated testbed implementation. Specifically, we employed the Pokemon Go AR mode to represent the VR/AR service, the Pokemon Go player versus player (PVP) mode for the online gaming service, and Twitch for the live video streaming service. Within our testbed setup, we employ two Intel i7-10700k CPU nodes connected to a 1Gb/s LAN. We run docker containers created from images based on Ubuntu 22.04 LTS, emulating the VMs within our system model. The overall workflow of our testbed is depicted in Fig. 3. In the experimentation process, we first executed

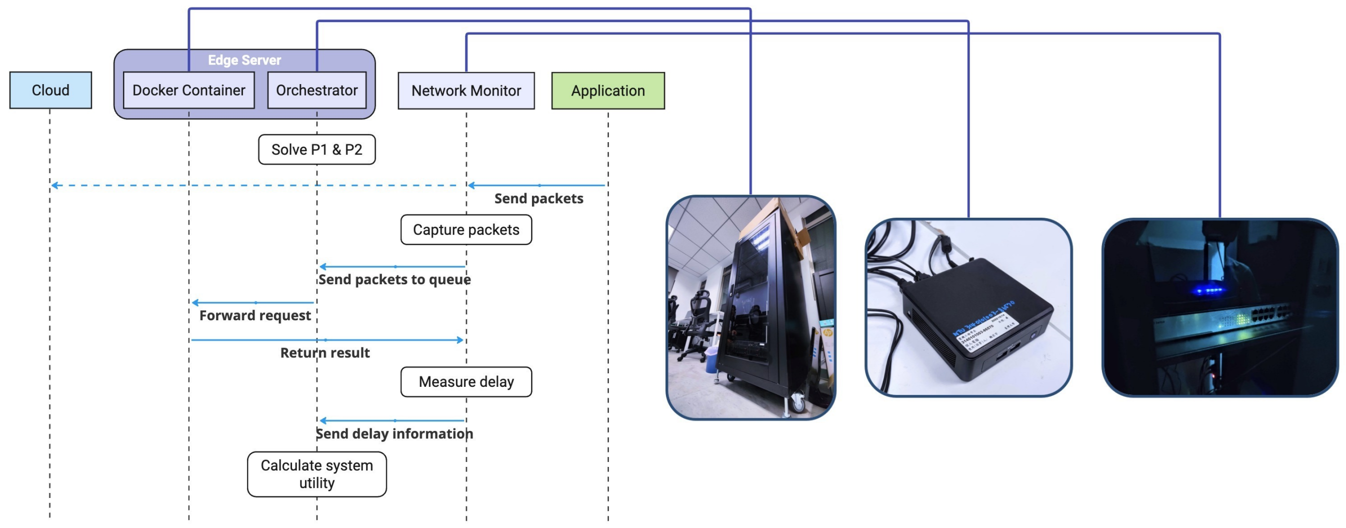
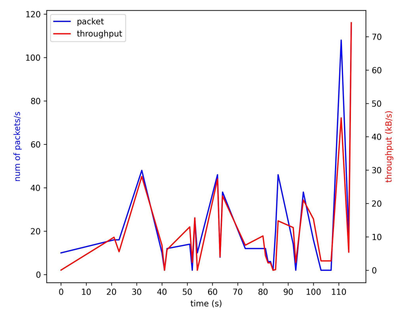
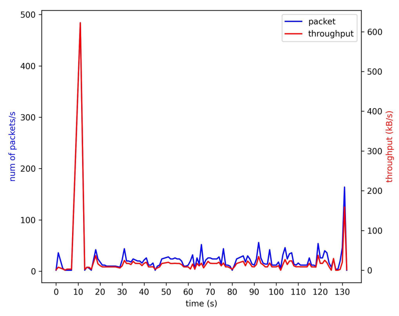


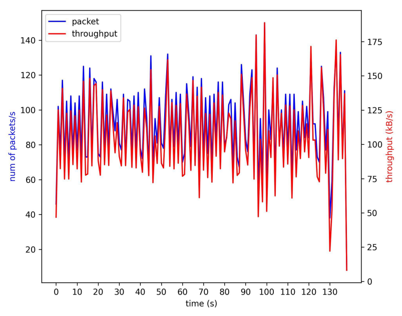
Fig. 3: The implemented testbed evaluation workflow.



(a) Pokemon AR mode traces



(b) Pokemon Go PVP model traces



(c) Twitch traces

Fig. 4: The captured network traces of actual applications.

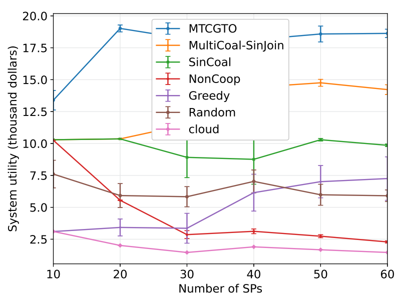
our algorithm using the simulation settings to achieve the final stable coalition formation. Subsequently, we launched Pokemon Go in AR mode, engaged in Pokemon Go PVP battles, and streamed content from Twitch on our mobile devices while connected to our edge network. We employed a network monitor to capture the packets generated during these activities, as illustrated in Fig. 4. These packets were then routed to the respective docker containers on our edge network based on the assignment established by the coalition formation. Each set of docker containers maintained a packet queue, and the packets were processed in a round-robin manner. Finally, after the packets were processed, the overall delay was measured and reported back to the orchestrator to calculate the system utility, which was then compared to the simulation results. It is worth mentioning that due to resource constraints and cost limitations of our physical equipment, we conducted smaller-scale experiments in our testbed.

Unless stated otherwise, the default parameter settings for N, M, S are provided as follows: $N = 30$, $M = 3$, $S = 450$. We compared the proposed MSCGTO scheme against the following six schemes:

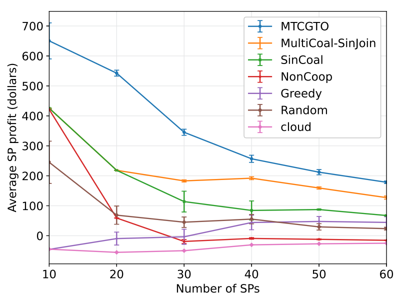
- `MultiCoal-SinJoin`: Proposed in our previous work [16], This scheme allows multiple coalitions on each edge server, but SPs can only join a single coalition. Each SP is assumed to have a static pool of SSs.
- `SinCoal`: Proposed in [14], this scheme enables SPs using the same edge server to decide whether to collaborate or not. If they choose to collaborate, they form a single coalition. Thus, there is at most one coalition for each edge server.
- `NonCoop`: In this scheme, each SP can only utilize its own leased edge server resources and does not engage in any collaborations.
- `Greedy`: Each SP greedily offloads its SS requests to the closest server.
- `Random`: Each SP randomly joins a coalition, without any strategic decision-making.
- `Cloud`: This scheme restricts SPs from offloading their tasks to the edge servers, requiring them to solely rely on the cloud for their service provisioning.

B. Impact of Profit from the Number of SPs in Simulations

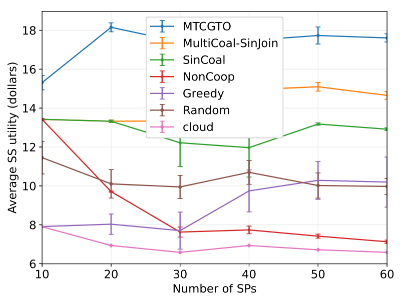
Fig. 5 shows how the total system utility changes with different numbers of SPs. We know that system utility is



(a) System utility

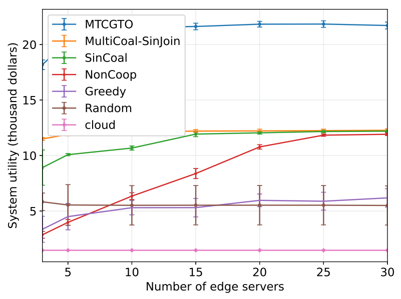


(b) Average SP profit

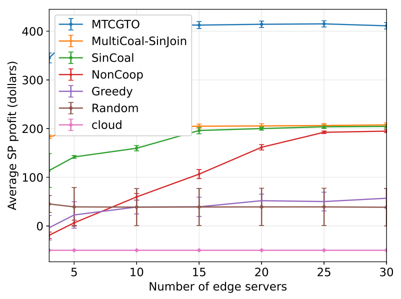


(c) Average SS utility

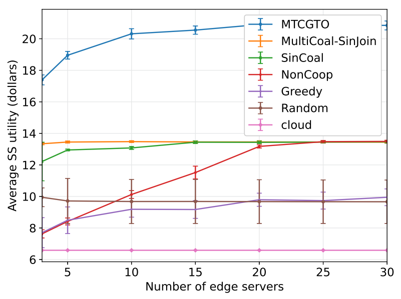
Fig. 5: Performance with different numbers of SPs in computer-based simulation.



(a) System utility

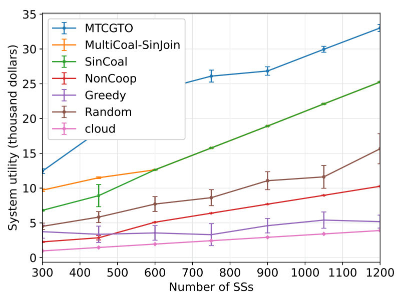


(b) Average SP profit

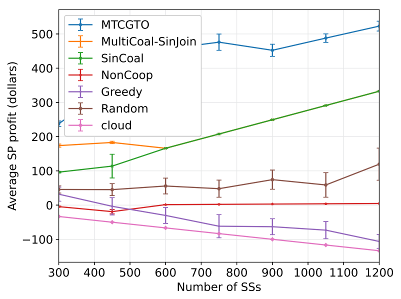


(c) Average SS utility

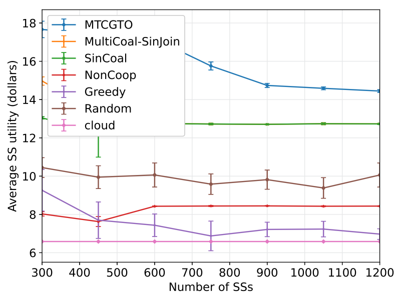
Fig. 6: Performance with different numbers of edge servers in computer-based simulation.



(a) System utility

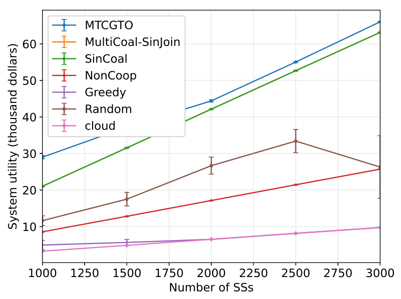


(b) Average SP profit

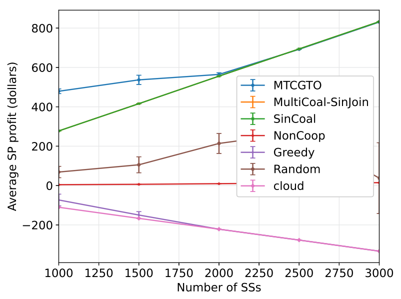


(c) Average SS utility

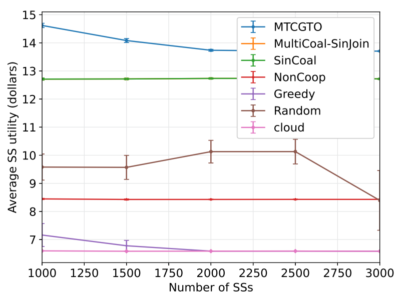
Fig. 7: Performance with different numbers of SSs in computer-based simulation.



(a) System utility



(b) Average SP profit

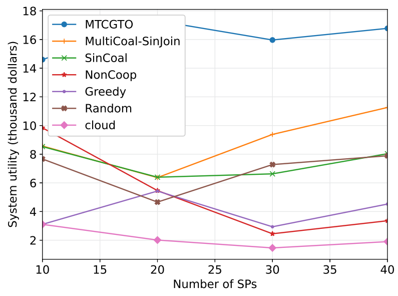


(c) Average SS utility

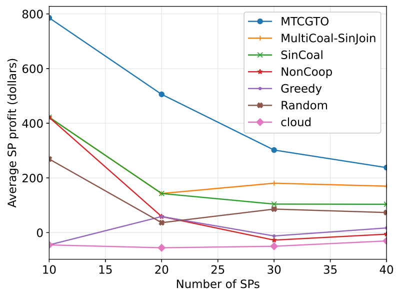
Fig. 8: Performance with large numbers of SSs in computer-based simulation.

composed of the total profit of SPs and the total utility of SSs, so we can analyze it from these two aspects.

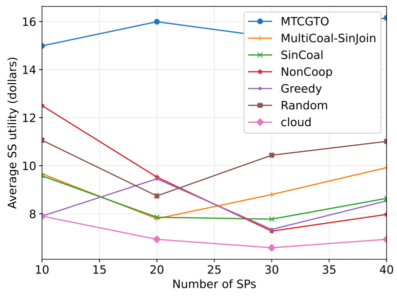
First, let's look at Fig. 5b. Since the number of SSs is held constant, as the number of SPs increases, the average number



(a) System utility

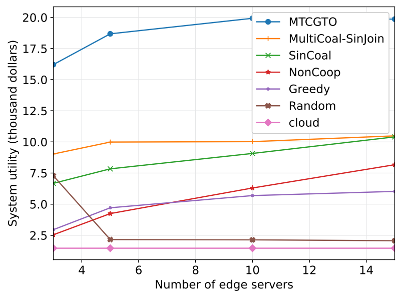


(b) Average SP profit

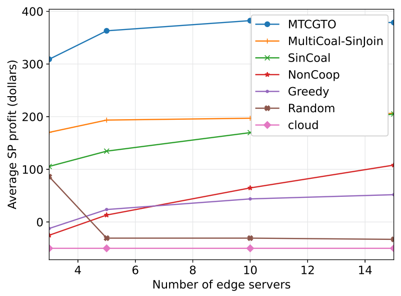


(c) Average SS utility

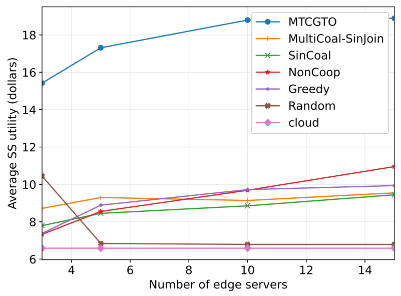
Fig. 9: Performance with different numbers of SPs in real-world testbed.



(a) System utility

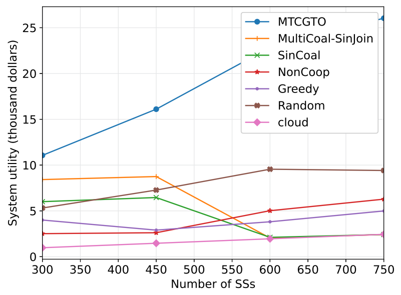


(b) Average SP profit

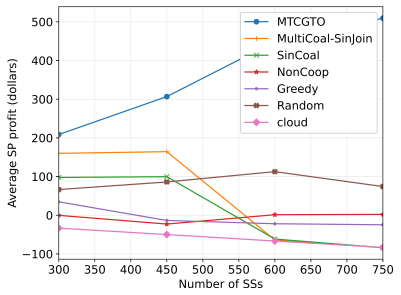


(c) Average SS utility

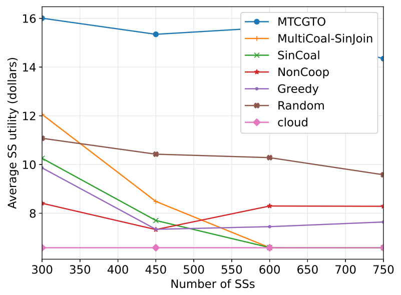
Fig. 10: Performance with different numbers of edge servers in real-world testbed.



(a) System utility



(b) Average SP profit



(c) Average SS utility

Fig. 11: Performance with different numbers of SSs in real-world testbed.

of SSs served by each SP decreases, leading to a decrease in the average profit of each SP.

Next, we assess the average utility of SSs. As illustrated in Fig. 5c, for the proposed `MSCGTO` and `MultiCoal-SinJoin` scheme, an increase in the number of SPs leads to more choices for each SS, resulting in a higher average utility for SSs. In contrast, for the `SinCoal` scheme, where only one coalition is allowed per edge server, the number of choices for both SPs and SSs is limited, resulting in no significant change in the average SS utility.

Overall, we observe that the `MSCGTO` scheme consistently achieves the highest utility among the evaluated algorithms.

C. Impact of Profit From the Number of Edge Servers in Simulations

Fig. 6 shows how the system utility changes with different numbers of edge servers. For `MultiCoal-SinJoin` and `SinCoal` schemes, an increase in the number of edge servers initially leads to a higher system utility. This is because having more edge servers provides a greater number of choices for SPs to offload their services and form coalitions, resulting in improved delay reduction and higher profit. However, after reaching a certain point, the increase in system profit becomes minimal. This is because most SSs have already achieved their desired delay requirements, and further reducing the delay does not significantly impact the SPs' profitability. In contrast, the proposed `MSCGTO` scheme does not exhibit the same trend. This is because `MSCGTO` is designed to efficiently allocate SSs to SPs, even with a limited number of edge servers. Therefore, even when there are fewer edge servers, the `MSCGTO` scheme can still satisfy the majority of SSs, leaving little room for further utility improvement.

D. Impact of Profit From the Number of SSs in Simulations

We now examine the impact of the number of SSs on total system utility, as displayed in Fig. 7. From Fig. 7b, we observe that the average utility of an SP increases with the number of SSs. This can be attributed to the fact that with more SSs, each SP serves a larger number of SSs on average, resulting in higher revenue and profitability. However, as depicted in Fig. 7c, the average utility of SSs decreases as the number of SSs increases. This can be attributed to the higher computation load when more SSs are present, leading to increased service delay for each SS. Consequently, the overall satisfaction and utility of SSs decline.

To investigate the scalability of the proposed scheme, we conducted additional experiments with larger numbers of SSs, as shown in Fig. 8. These experiments revealed a consistent trend, echoing the findings from smaller-scale scenarios. Notably, the `MSCGTO` approach consistently maintains the highest average SS utility, which can be largely attributed to the relatively low latency it achieves in the VR service when compared to alternative approaches. These results underscore the exceptional scalability of the `MSCGTO` framework.

E. Testbed Results Evaluation

We conducted a smaller-scale experiment on our testbed using real data to validate the performance of the proposed scheme. The experimental results are presented in Figs. 9 to 11, showcasing similar trends as observed in the simulations. One notable difference is observed in the performance of `MultiCoal-SinJoin` and `SinCoal` when there is a high number of SSs, as depicted in Fig. 11. Due to the hardware limitations of our testbed, when the number of SSs increases significantly, the real delay experienced by them becomes excessively high. As a consequence, the profit of SPs and the utility of SSs collapse in these schemes. In contrast, the proposed `MSCGTO` method allows each SP to distribute its SS requests among multiple coalitions, effectively reducing the load on each coalition. By constraining latency within the bounds of SS requirements, the system is able to benefit from the increase in subscribers, leading to improved profitability for SPs and increased utility for SSs, thereby achieving superior scalability in a complex real-world environment.

F. Algorithms Evaluation

Based on the 95% confidence intervals of the simulation results, as presented in Figs. 5 to 7, it becomes clear that `MSCGTO` exhibits a statistically significant advantage over other existing algorithms. This observation is further supported by the testbed results showcased in Figs. 9 to 11, providing additional evidence of the practical effectiveness of the proposed scheme. The `MSCGTO` approach consistently achieves the highest system utility across various scenarios, including reaching a 25% increase in system utility compared to `MultiCoal-SinJoin` [16] and an 87.5% increase compared to `SinCoal` [14] when there's a large number of SPs in simulation as shown in Fig. 5, demonstrating better scalability. This can be attributed to two key factors. Firstly, it allows SPs to participate in multiple coalitions across different edge servers, enabling them to effectively serve geographically distributed SSs and reduce propagation delay. SPs can also effectively balance their workload by serving SSs from multiple coalitions, reducing queuing delays, and ensuring efficient utilization of resources. Secondly, it facilitates SSs in dynamically switching between SPs to minimize their individual latency, thereby elevating the overall system performance and user satisfaction. By leveraging these features, `MSCGTO` is able to deliver superior system utility and provide an effective solution to address the challenges in distributed service provisioning.

VII. Conclusion

In this paper, we focused on addressing the resource allocation optimization problem in a multi-service system characterized by geographically distributed service subscribers (SSs) and the ability of service providers (SPs) to offload tasks to multiple edge servers. Furthermore, SSs are granted the freedom to switch between SPs based on their preferences and requirements. We propose a multi-stage coalition game task offloading (MSCGTO) framework to tackle the problem. To bridge the gap between theoretical analysis and practical implementation, we conducted experiments on a real-world testbed using real data from VR/AR, online gaming, and live video streaming services. The results demonstrated the effectiveness and scalability of the proposed algorithms in contrast to existing approaches in both simulation and real-world testing. Specifically, the MSCGTO framework consistently outperformed all other evaluated algorithms, yielding a significant increase of 25% in system utility, considering both SP profitability and SS latency, compared to our previous research and an 87.5% increase compared to another work in large-scale scenarios. In our future research, we will explore the intricacies of privacy by design, security, risk management, and the incorporation of heterogeneous VMs to further enhance the practicality and robustness of our proposed framework. Additionally, we will investigate the applicability of Stackelberg games and reinforcement learning for more accurate modeling of interactions between SPs and SSs.

References

[1] Ericsson, "Ericsson mobility report," Nov. 2022. [Online]. Available: https://www.ericsson.com/4ae28d/assets/local/reports-papers/mobility-report/documents/2022/ericsson-mobility-report-november-2022.pdf

[2] ETSI, "Mobile edge computing (mec); framework and reference architecture," ETSI, DGS MEC, standard 3, 2016.

[3] T.-Y. Chen, Y. Chiang, J.-H. Wu, H.-T. Chen, C.-C. Chen, and H.-Y. Wei, "Ieee p1935 edge/fog manageability and orchestration: Standard and usage example," in *2023 IEEE International Conference on Edge Computing and Communications (EDGE)*, Jul. 2023, pp. 96–103.

[4] X. Wang, M. Chen, T. Taleb, and A. Ksentini, "Cache in the air: Exploiting content caching and delivery techniques for 5g systems," *Communications Magazine, IEEE*, vol. 52, pp. 131–139, Feb. 2014.

[5] T.-Y. Kan, Y. Chiang, and H.-Y. Wei, "Qos-aware mobile edge computing system: Multi-server multi-user scenario," in *2018 IEEE Globecom Workshops (GC Wkshps)*, 2018, pp. 1–6.

[6] ——, “Task offloading and resource allocation in mobile-edge computing system,” in *2018 27th Wireless and Optical Communication Conference (WOCC)*, 2018, pp. 1–4.

[7] Y. Chiang, Y. Zhang, H. Luo, T.-Y. Chen, G.-H. Chen, H.-T. Chen, Y.-J. Wang, H.-Y. Wei, and C.-T. Chou, “Management and orchestration of edge computing for iot: A comprehensive survey,” *IEEE Internet of Things Journal*, vol. 10, no. 16, pp. 14 307–14 331, 2023.

[8] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, “Mobile edge computing: A survey,” *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2018.

[9] P. Ranaweera, A. D. Jurcut, and M. Liyanage, “Survey on multi-access edge computing security and privacy,” *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 1078–1124, 2021.

[10] T. X. Tran and D. Pompili, “Octopus: A cooperative hierarchical caching strategy for cloud radio access networks,” in *2016 IEEE 13th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, 2016, pp. 154–162.

[11] A. Mehrabi, M. Siekkinen, and A. Ylä-Jaaski, “Qoe-traffic optimization through collaborative edge caching in adaptive mobile video streaming,” *IEEE Access*, vol. 6, pp. 52 261–52 276, 2018.

[12] Y. Chiang, C.-H. Hsu, and H.-Y. Wei, “Collaborative social-aware and qoe-driven video caching and adaptation in edge network,” *IEEE Transactions on Multimedia*, vol. 23, pp. 4311–4325, 2021.

[13] Z. Xu, L. Zhou, S. Chi-Kin Chau, W. Liang, Q. Xia, and P. Zhou, “Collaborate or separate? distributed service caching in mobile edge clouds,” in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020, pp. 2066–2075.

[14] Z. Xu, L. Zhou, S. C.-K. Chau, W. Liang, H. Dai, L. Chen, W. Xu, Q. Xia, and P. Zhou, “Near-optimal and collaborative service caching in mobile edge clouds,” *IEEE Transactions on Mobile Computing*, vol. 22, no. 7, pp. 4070–4085, 2023.

[15] R. Wu, G. Tang, T. Chen, D. Guo, L. Luo, and W. Kang, “A profit-aware coalition game for cooperative content caching at the network edge,” *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 1361–1373, 2022.

[16] C.-C. Lin, Y. Chiang, and H.-Y. Wei, “Collaborative edge caching with multiple virtual reality service providers using coalition games,” in *2023 IEEE Wireless Communications and Networking Conference (WCNC)*, 2023, pp. 1–6.

[17] Z. Su and Q. Xu, “Security-aware resource allocation for mobile social big data: A matching-coalitional game solution,” *IEEE Transactions on Big Data*, vol. 7, no. 4, pp. 632–642, 2021.

[18] S. Luo, X. Chen, Z. Zhou, X. Chen, and W. Wu, “Incentive-aware micro computing cluster formation for cooperative fog computing,” *IEEE Transactions on Wireless Communications*, vol. 19, no. 4, pp. 2643–2657, 2020.

[19] N. Zhao, H. Wu, and Y. Chen, “Coalition game-based computation resource allocation for wireless blockchain networks,” *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8507–8518, 2019.

[20] L. Chen, C. Shen, P. Zhou, and J. Xu, “Collaborative service placement for edge computing in dense small cell networks,” *IEEE Transactions on Mobile Computing*, vol. 20, no. 2, pp. 377–390, 2021.

[21] A. Asheralieva and D. Niyato, “Hierarchical game-theoretic and reinforcement learning framework for computational offloading in uav-enabled mobile edge computing networks with multiple service providers,” *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8753–8769, 2019.

[22] Q.-V. Pham, H. T. Nguyen, Z. Han, and W.-J. Hwang, “Coalitional games for computation offloading in noma-enabled multi-access edge computing,” *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 1982–1993, 2020.

[23] M. Chen and Y. Hao, “Task offloading for mobile edge computing in software defined ultra-dense network,” *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587–597, 2018.

[24] I. A. Elgendy, W.-Z. Zhang, Y. Zeng, H. He, Y.-C. Tian, and Y. Yang, “Efficient and secure multi-user multi-task computation offloading for mobile-edge computing in mobile iot networks,” *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2410–2422, 2020.

[25] T. X. Tran and D. Pompili, “Joint task offloading and resource allocation for multi-server mobile-edge computing networks,” *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 856–868, 2019.

[26] Y. Chiang, C.-H. Hsu, G.-H. Chen, and H.-Y. Wei, “Deep q-learning-based dynamic network slicing and task offloading in edge network,” *IEEE Transactions on Network and Service Management*, vol. 20, no. 1, pp. 369–384, 2023.

[27] G. Mitsis, E. E. Tsiropoulou, and S. Papavassiliou, “Price and risk awareness for data offloading decision-making in edge computing systems,” *IEEE Systems Journal*, vol. 16, no. 4, pp. 6546–6557, 2022.

[28] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, “In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning,” *IEEE Network*, vol. 33, no. 5, pp. 156–165, 2019.

[29] S. Müller, O. Atan, M. van der Schaar, and A. Klein, “Context-aware proactive content caching with service differentiation in wireless networks,” *IEEE Transactions on Wireless Communications*, vol. 16, no. 2, pp. 1024–1036, 2017.

[30] “IEEE standard for edge/fog manageability and orchestration,” *IEEE standard 1935-2023*, 2023.

[31] M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

[32] W. Saad, Z. Han, M. Debbah, A. Hjorungnes, and T. Basar, “Coalitional game theory for communication networks,” *IEEE Signal Processing Magazine*, vol. 26, no. 5, pp. 77–97, 2009.

[33] O. A. Wahab, J. Bentahar, H. Otrok, and A. Mourad, “Towards trustworthy multi-cloud services communities: A trust-based hedonic coalitional game,” *IEEE Transactions on Services Computing*, vol. 11, no. 1, pp. 184–201, 2018.

[34] A. Bogomolnaia and M. O. Jackson, “The stability of hedonic coalition structures,” *Games and Economic Behavior*, vol. 38, no. 2, pp. 201–230, 2002.

[35] S. Mangiante, G. Klas, A. Navon, G. Zhuang, J. Ran, and M. Silva, “Vr is on the edge: How to deliver 360° videos in mobile networks,” Aug. 2017, pp. 30–35.

[36] K. Raaen and A. Petlund, “How much delay is there really in current games?” in *Proceedings of the 6th ACM Multimedia Systems Conference*, ser. MMSys ’15, New York, NY, USA, 2015, p. 89–92.

[37] C. Zhang and J. Liu, “On crowdsourced interactive live streaming: A twitch.tv-based measurement study,” *Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video, NOSSDAV 2015*, Feb. 2015.

Chun-Che Lin received the B.S. degree in Electrical Engineering from National Taiwan University (NTU), Taiwan, in 2023. His research interests include edge computing, resource allocation, and game theory.

Yao Chiang received the B.S. degree and the M.S. degree in management information systems from National Pingtung University of Science and Technology (NPUST), Pingtung, Taiwan, in 2014 and 2016, respectively. He received the Ph.D. degree in electrical engineering at National Taiwan University (NTU) in 2021, and he is currently a postdoctoral research fellow in NTU. His research interests include data mining, machine learning, and mobile communications design for multi-access edge computing systems.

Hung-Yu Wei Hung-Yu Wei is a Professor in Department of Electrical Engineering and Graduate Institute of Communications Engineering, National Taiwan University. Currently, he serves as Associate Chair in Department of Electrical Engineering. He received the B.S. degree in electrical engineering from National Taiwan University in 1999. He received the M.S. and the Ph.D. degree in electrical engineering from Columbia University in 2001 and 2005 respectively. He was a summer intern at Telcordia Applied Research in 2000 and 2001. He was with NEC Labs America from 2003 to 2005. He joined Department of Electrical Engineering at the National Taiwan University in July 2005. His research interests include next-generation wireless broadband networks, IoT, vehicular networking, fog/edge computing, cross-layer design and optimization in wireless multimedia communications, and game theoretical models for communications networks.

Dr. Wei received NTU Excellent Teaching Award in 2008 and 2018. He also received ”Recruiting Outstanding Young Scholar Award” from the Foundation for the Advancement of Outstanding Scholarship in 2006, K. T. Li Young Researcher Award from ACM Taipei/Taiwan Chapter and The Institute of Information and Computing Machinery in 2012, Ministry of Science and Technology Research Project for Excellent Young Scholars in 2014, Excellent Young Engineer Award from the Chinese Institute of Electrical Engineering in 2014, Wu Ta You Memorial Award from MOST in 2015, and Outstanding Research Award from MOST in 2020. He was a consulting member of Acts and Regulation Committee of National Communications Commission during 2008 2009. He served as a division director in NTU Computer and Information Networking Center during 2016-2017. He has been actively participating in NGMN, IEEE 802.16, 3GPP, IEEE P1934, and IEEE P1935 standardization. He serves as Vice Chair of IEEE P1934 Working Group to standardize fog computing and networking architecture. He serves as Secretary for IEEE Fog/Edge Industry Community. He also serves as an Associate Editor for IEEE IoT journal. He is an IEEE certified Wireless Communications Professional. He was the Chair of IEEE VTS Taipei Chapter during 2016 2017. He is currently the Chair of IEEE P1935 working group for edge/fog management and orchestration standard.