Repeated Auction Mechanisms for Multi-Access Edge Computing

by

Ummy Habiba

A Thesis submitted to The Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Electrical and Computer Engineering
University of Manitoba
Winnipeg

December 2022

Abstract

Mobile edge computing (MEC) is one of the promising technologies that ensures high data rate and ultra low service latency with the computational capabilities at the edge of 5G wireless/cellular networks and beyond. Due to the scarcity of the computing resources at the edge servers, it is crucial to develop an efficient resource allocation mechanism that benefits both the resource sellers and offloading mobile users with heterogeneous QoE requirements. Despite extensive studies on MEC offloading, existing research lacks efficient resource allocation mechanism addressing the stochastic nature in resource demands as well as computational capacities of the servers. Besides, there is a significant research gap reflecting the economical efficiency in a computation offloading service market, which supports the growing market size of MEC-enabled applications, and the competition among different business rivals, e.g., mobile network operator, wireless or computing service providers, etc. In a competitive market scenario, the auction game theory has been widely popular for designing efficient resource allocation mechanisms, as it particularly focuses on regulating the strategic interactions among the self-interested players. In this thesis, I investigate auction-based approaches to model a dynamic MEC offloading service market model, addressing the resource allocation problem with the goal of ensuring consistent QoE for offloading users as well as maximizing the auction revenue. To achieve this research goal, I develop repeated auction mechanism considering the network dynamics in computation offloading, and design a novel generalized second price (GSP) mechanism to obtain efficient offloading task assignment and resource allo-

cation pricing decisions. Furthermore, I study adaptive best-response bidding strategies that maximize the profits of the resource sellers, and guarantee the stability and effectiveness of the auction by satisfying desired economic properties. To this end, I validate the performance of the proposed repeated auction mechanisms and bidding strategies through numerical result analysis.

Table of Contents

List of Tables

List of Figures

Abbreviations

5G Fifth-generation.

AB Altruistic Bidding.

AF Application Function.

API Application Programming Interface.

AR Augmented Reality.

BB Balanced Bidding.

CB Competitor Busting.

CE Computational Efficiency.

CPU Central Processing Unit.

CR Computing Resource.

CSP Computation Service Provider.

CTR Click-through-rate.

EGW Edge Gateway.

eNB Evolved Node B.

ETSI European Telecommunications Standards Institute.

GFP Generalized First Price.

GSP Generalized Second Price.

IC Incentive Compatibility.

IoT Internet-of-Things.

IR Individual Rationality.

ISG Industry Specification Group.

LEFE Locally Envy-Free Equilibrium.

LTE Long-Term Evolution.

MCC Mobile Cloud Computing.

MDMCK Multi-dimensional Multiple Choice Knapsack Problem.

MEC Mobile Edge Computing / Multi-access Edge Computing.

MNO Mobile Network Operator.

NE Nash Equilibrium.

NFV Network Function Virtualization.

OTT Over-the-Top.

QoE Quality-of-Experience.

QoS Quality-of-Service.

RAN Radio Access Network.

RBB Restricted Balanced Bidding.

SDN Software-Defined Network.

SNE Symmetric Nash Equilibrium.

SP Service Provider.

UAV Unmanned Aerial Vehicle.

UE User Equipment.

UPF User Plane Function.

VCG Vickrey-Clarke-Groves.

vCPU Virtual CPU.

VM Virtual Machine.

VR Virtual Reality.

WAP Wireless Access Point.

WDP Winner Determination Problem.

WR Wireless Resource.

Chapter 1

Introduction

The exponential growth of mobile internet traffic and the rising popularity of diverse high-performance wireless applications drive the enormous advancements in wireless applications in 5G networks and beyond. To satisfy users' growing demand, the wireless communication technologies have advanced through numerous intelligent applications such as virtual reality (VR), augmented reality (AR), IoT data analytic, autonomous vehicles, UAV/drone-based communication, advanced social networking, etc. All these intelligent applications give rise to the requirements for higher computing efficiency, real-time communication, and ubiquitous network connectivity. As a result, the network edge has been getting crowded with large number of requests for computational capabilities from various types of mobile or IoT smart devices.

This large-scale deployment of intelligent wireless applications involves numerous computationally-intensive and latency-critical tasks. Although advanced smart devices possess a significant computational processing capacity, they suffer from limited battery life. Besides, traditional cloud computing model where computationally-intensive and latency-critical tasks are forwarded to the centralized cloud computing

data center is not a viable solution due to long propagation delays and degradation of quality-of-service (QoS). Hence, the Mobile Edge Computing, also called Multi-access Edge Computing (MEC) has emerged, which offers higher computation performance computing capabilities (e.g., data, computation, and storage solutions) at the edge of cellular network in compare to the existing Mobile Cloud Computing (MCC) [44] architecture. In MEC networks [45], the computing servers are deployed within the cellular radio access network (RAN), which allows end users to offload their computation-intensive and delay-sensitive computation tasks to nearby edge servers instead of forwarding them to the remote cloud server. Thus, communication and processing delay with increased energy consumption due to spatial distance between the mobile devices and cloud servers are significantly reduced in MEC. MEC enhances users' quality of experience (QoS) and also provides them with high bandwidth, ultra-low latency, reduced energy consumption, lower offloading cost, real-time and location-aware services [47].

In this chapter, we first introduce MEC network architecture in section 1.1, detailing the concepts of computation offloading and dynamic resource allocation for MEC using the SDN functionalities. In section 1.2, I discuss the motivation of this thesis, outlining the challenges in implementing MEC offloading in practice, stating the research problems I investigate in this thesis, and the methodology to address the problems. The state-of-the-art of related work on MEC offloading mechanisms is discussed in section 1.3. I summarize the contributions of this thesis in section 1.4 and finally conclude the chapter in section 1.5 by presenting a snapshot of each chapter in this thesis.

1.1 Multi-Access Edge Computing (MEC)

Multi-Access Edge Computing (MEC) is one of the key emerging technologies that enables the implementation Network Functions Virtualization (NFV) and Software-Defined Networking (SDN)) functionalities in 5G wireless networks and beyond. The MEC refers to the computing platform, where computing servers are located in close proximity of data sources, i.e., user equipment (UEs) or mobile devices. The computation-intensive tasks, which are requested by the UEs to be at the MEC platform, are executed by harvesting idle computing resources and storage from the edge servers. Thus, the end-to-end communication delay is reduced significantly in compare to cloud computing paradigm where computing tasks are forwarded to the remote data centers.

The main difference between the MEC and cloud computing architectures is that MEC is primarily focused on the cellular/wireless data network instead of general internet. MEC avoids internet data transmission between users and cloud servers, by deploying the computing servers at the wireless access points (WAPs) within the radio access network (RAN), e.g., base stations, radio network controllers, or multi-technology cell aggregation sites [38]. Thus, MEC manages to provide computing services with ultra-low latency and offers highly reliable, bandwidth efficient and secure connections, so that wireless service or infrastructure providers can serve their customers with high quality-of-experience (QoE). Therefore, MEC has become the prominent choice to provision emerging wireless applications, which include but not limited to over-the-top multi-media streaming services [5], online interactive games [6], augmented-and virtual reality, and tactile internet [50],

video analytics [57], smart city [52] and healthcare services [3], automated vehicular applications [63], factory automation using industrial internet of things (IIoT) devices [18].

The MEC concept enables the wireless service- and infrastructure providers to access to heterogeneous fixed and mobile wireless access technologies in WiMax, 4G/LTE, 5G networks and beyond. The SDN functionalities facilitates the integration of MEC with the existing 3GPP network architecture without making significant changes to the hardware infrastructure specifications [34]. Recently, the Industry Specification Group (ISG) within European Telecommunications Standards Institute (ETSI) has released new specifications on MEC 5G integration, detailing the application programming interfaces (APIs) which deploys the MEC platform as an application function (AF) for interacting with the 5G system [33]. Hence, MEC can be easily deployed within the RAN, by installing the a site-controller at WAPs, routers or gateways, which bascially host the MEC service APIs. The MEC site-controller can also be deployed in central locations like data centers of network operators or on moving nodes like passenger vehicles or UAVs. As such, the MEC system can utilize local radio-network contextual information to guarantee secure, reliable, and privacy-preserving services based on intelligent analysis and data processing capabilities at the edge [3].

With the integration of MEC into the wireless networks, new business opportunities emerge where mobile network operators (MNOs) can capitalize on the computation capabilities of MEC by opening up their networks to authorized third-parties, independent software ven-

dors, over-the-top (OTT) market players, and application developers. In a SDN-assisted MEC system, a single MEC node can with multiple 5G network operators, and vice versa; through service specific APIs. This creates a multi-vendor MEC market, where a service provider (i.e., vendor) can deploy its application utilizing the MEC service delivery platform and serve his/her customers belonging to different network operators. Hence, rivalry arise when a number of entities, ranging from MNOs to mobile application developers, compete with each other to access MEC infrastructure resources for their service delivery.

1.1.1 Computation Offloading in MEC

Computation offloading is one of the popular applications of MEC, where UEs get to execute their computation intensive or delay sensitive tasks at a nearby edge server instead of execute them at their local CPUs. Computation offloading not only speeds up the task execution process but also saves energy which improves users' QoE significantly. However, computation offloading involves several critical decision making at the user ends, such as : (a) whether to offload to a server or compute locally, (b) which task to offload, and (c) how much data to offload for each task. The user decides to offload if he/she finds the cost of offloading is cheaper than running it by itself at the local processor. Generally, there are two offloading scenarios [31]:

- **Full Offloading**: The entire computing task of an application is offloaded and executed at the MEC.
- **Partial Offloading**: A part of the computing task is offloaded and computed at the MEC, and the rest is processed locally.

When the end users decides to offload to MEC (either full or partial offloading), the further task processing decisions are taken care at the MEC platform is taken care either in centralized or decentralized manner [31]. In this research, I explore the prospects of provisioning computation offloading as a service in a multi-vendor MEC market environment. Hence, in order to enforce regulated offloading service trading between the computing resource sellers and offloading users, I consider a software-defined centralized service broker which is referred to as the *MEC orchestrator* throughout the thesis.

A deployment scenario for computation offloading in a multi-vendor MEC network architecture is presented in Fig. 1.1, where the network components are divided into the following tiers:

- **Tier 1:** The first tier comprises several heterogeneous 5G wireless application devices within the access network. Each device, referred to as user equipment (UE), is compatible with LTE and

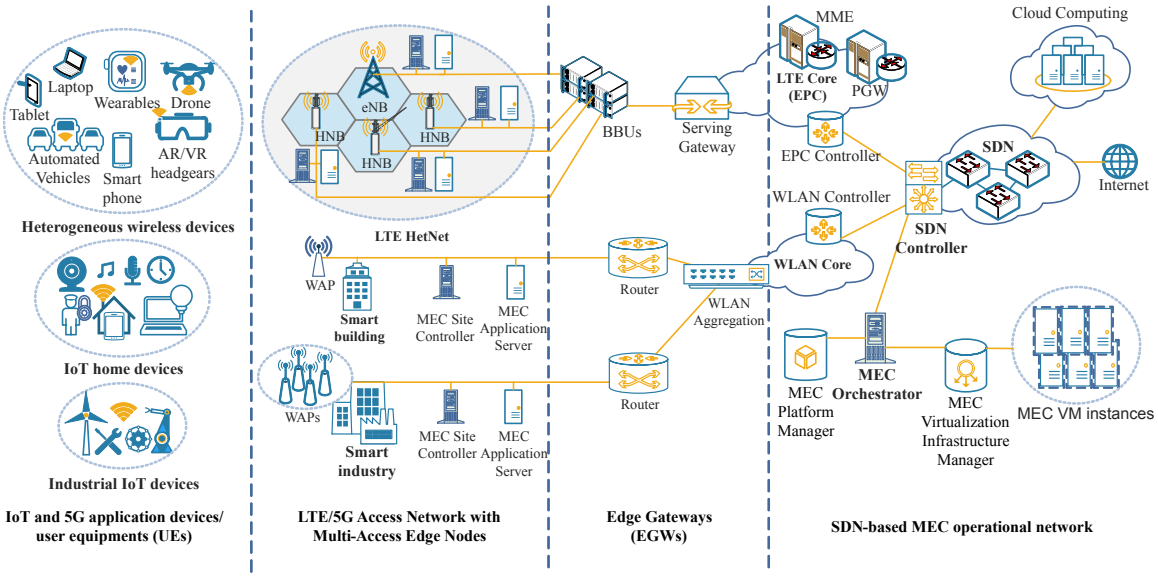


Figure 1.1: SDN-based multi-tier multi-access edge computing infrastructure and communication network architecture.

WiFi wireless communications standards. Examples include smartphones, AR/VR devices, drones, and IoT devices.

- **Tier 2:** The second tier includes distinct wireless access nodes such as eNodeB (eNB) within the LTE and WLAN access networks. These wireless access points (WAPs) are the MEC nodes/sites deployed onto cellular base stations, buildings, and vehicles. Physical servers and site controllers locate within the premises of the WAPs and gather the computation offloading requests from UEs via the associated WAPs.

- **Tier 3:** In this tier, the computation offloading requests received by the MEC nodes are forwarded to the edge gateways (EGWs) using wireless routers and aggregation switches. The offloading requests are pre-processed at the EGWs and then directed to the MEC system-level controller node in the core network tier.

- **Tier 4:** The core tier is enabled with software-defined network (SDN) capabilities, where a central unit coordinates all the operational functionalities of LTE, WLAN, and MEC via different controller nodes. Due to the scarcity of resources, the servers might be unable to process all of the offloaded tasks. In that case, they forward some of them to the cloud computing platform and central data centers. In our model, a broker or hypervisor, namely, the MEC orchestrator, manages all the MEC components.

1.1.2 Resource Allocation for MEC Offloading Services

In a service-oriented MEC system architecture, computation offloading process can be deployed as a service, where the orchestrator partitions

available resources, and allocates a certain partition/slice from one of the edge servers to a particular UE. A monetary amount, as the payment for utilizing the allocated resources, is then paid by the UE to the service provider who owns the server via the orchestraor. The orchestrator is in-charge of overall computation offloading service provisioning process, which include: (a) receiving offloading requests, (b) determining offloading task assignment decision, and (c) allocating computing resources from the server to the assigned tasks, and (d) deciding the offloading service price for the allocated resources. The orchestrator interacts with the MEC sites, EGWs and UEs through control and user plane functions (UPFs), and implements the offloading services with the help of two other SDN controllers: (a) MEC virtualization infrastructure manager and (b) MEC platform manager.

The MEC virtualization manager mainly oversees the computing infrastructure resources (i.e., servers) of the MEC system. It abstracts and partitions the computing resources into virtualized CPU (vCPU) resource units, using SDN and NFV functionalities. The virtualization manager defines a set virtual machine (VM) instances for each server, by assigning vCPUs as required to meet the computation processing requirements of each application type. The virtualization manager thus maintains a resource pool, which consists of VM resources available at edge server, and monitors their computational workloads, CPU performance and resource utilization metrics. The MEC platform manager supervises the offloading task execution process, by managing the separate task processing queues for different wireless service/applications. The platform manager tracks the overall task execution process, start-

ing from arrival at the processing queue, to the assignment of a task to a server and exiting the queue after task computation is finished.

The orechestrator filters each incoming offloading request by the requested application type, and then adds the task to the respective processing queue. The orchestrator matches each offloading task to a suitable VM which satisfies the QoE of the requesting UE, and maps the VM to the offloading task to compute the offloaded data. After the assigned VM finishes computing the task, the orchestrator collects the payment from the UEs and forwards it to the respective service provider. A single VM can be mapped to multiple tasks, sharing the computing resources by all the assigned task. This model would be suitable for offloading tasks of smaller data sizes. Conversely, a single offloading task can be matched to more than one VM. In such a case, the offloading data is computed in parts over the assigned VMs. This model helps process the delay-sensitive computation intensive tasks faster. In both model, the maximum number of VMs that can be matched to a computation task or the maximum number of tasks that a VM can process simultaneously, is limited and is defined by the vendor or service provider. For simplicity, I consider one-to-one matching for offloading task assignment decisions in this thesis. Detailed MEC service provisioning model is described in Chapter 4.

1.2 Motivation

The main motivation of this thesis to address the research challenges related to the large scale deployment of MEC-based applications in 5G wireless networks and beyond. Therefore, it is crucial to develop an

efficient resource management policies, so that the growing demand for computation offloading can be met using the limited resources at the MEC servers. In practice, the allocation of computing resources to implement MEC-based services is associated with several challenges, as I outline in the following section.

1.2.1 Research Challenges

One of the driving factors behind the advent of MEC technology is to minimize the end-to-end task execution latency and energy consumption at the UEs. Hence, the primary objective of any MEC service delivery platform is to provide UEs with high quality service at a minimal cost. In case of wireless networks, it becomes more challenging to meet UEs' QOS requirements due to the randomness, e.g., time-varying radio channel conditions, mobility of UEs, stochastic arrival of offloading requests, uncertainties in the task lengths, etc. Apart from that, there exists high level of competition among the UEs, over the limited resources at the servers, in a ultra-dense network setup. So, it becomes difficult to determine the optimal matching pairs between the offloading tasks and server, such UEs' QoS requirements are satisfied using the limited computing resources.

Besides, the computational workloads of the MEC servers are highly variable, due to varying length of computing tasks in an online computation offloading scenario. Although existing research considers the heterogeneity in resource demands and offloading task sizes, the dynamics in the workload-based computational performances of the servers are overlooked. Despite having, high configuration computation resources an overloaded server may not be able to finish the assigned tasks within

the task completion deadlines. Hence, the efficiency of MEC offloading resource management policy not only depends on the optimality of the offloading task assignment decisions, but also the dynamics of servers' QoS performance metrics.

Addressing the competition among UEs and limited resource capacity constraints at the servers, MEC resource allocation problem has been extensively studied in literature. Existing researches mostly focus on maximizing UEs' QoE in terms of offloading cost and task execution latency. Some recent works consider maximizing the social welfare, i.e., the sum utility of UEs and servers, and maximizing the profits of the MEC service provider or network operator. However, these studies mostly assume a single MEC service provider and/or network operator without addressing the competition when a number of profit-driven computing service providers may coexist in the same service location. Therefore, it is imperative to develop a standard MEC ecosystem and value chain, so that various new generation wireless service providers are motivated to deploy their services using the MEC platform. To do so, the MEC service delivery platform requires an efficient resource allocation and pricing mechanism that monetarily benefits both the UEs (i.e., resource buyers), and computing service providers (i.e., resource sellers).

Towards developing an efficient resource allocation mechanism for MEC offloading, the existing research investigates several distributed game- and optimization-based approaches [11, 62, 15, 30, 39]. However, auction theory appear as a popular choice in cutting-edge research, to study strategic interactions between rational entities in a competitive

market scenario. Especially, auction-based resource allocation and pricing strategies in dynamic computation offloading settings, are proven to be efficient, due to its inherent economic properties. Moreover, auction theory assists to implement fairness in service pricing, satisfying the demand-supply equilibrium criteria in the dynamic allocation of computing resources in MEC. Nevertheless, several challenges remain unaddressed in cutting edge research; particularly in designing competitive pricing strategies for computation offloading auction process in the MEC platform. The self-interested MEC service providers tend to apply their own bidding strategies to maximize their own profits. So, it is highly likely that bidders may misreport the bid prices, intending to manipulate the auction outcomes. In the dynamic auction setting, where service providers get the opportunity to learn other competitors offloading prices, it becomes harder to maintain the market equilibrium preventing the bidders from overbidding. Hence, it is crucial to study optimal bidding strategies so that such vindictive behavior can be prohibited, preserving the stability of the auction mechanism.

1.2.2 Research Problem

In this thesis, I study the resource allocation and pricing problem, considering an online auction-based MEC offloading service delivery platform, with multiple users and multiple computing service providers. The main objective of my research is to develop an economically efficient resource allocation mechanism, addressing the competition among the different service providers. Towards this goal, I investigate the following problems:

1. determine optimal resource allocation and pricing decisions for MEC offloading, such that total revenue of the service providers is maximized, under the constraint of satisfying users' heterogeneous resource demands.

2. determine optimal offloading service pricing strategies, that benefits both the users and self-interested service providers, and achieves a stable resource allocation solution under the dynamic MEC network condition.

In order to address these problems, I study the optimization approaches, particularly auction game theory approaches. In the following section, I discuss auction-related works on MEC offloading and potential dynamic resource allocation approaches to solve the above mentioned problems.

1.3 Related Work

The state-of-the-art in auction-based MEC offloading mechanism explores different approaches, addressing the following decision problems: (i) Task-server association, (ii) Resource allocation/provisioning for task-server pairs, and (iii) Computational resource pricing. In the MEC offloading auction framework, the task assignment- and resource allocation problems are jointly addressed as the winner determination problem. The winner is then solved using methods from convex optimization, mixed-integer programming, dynamic programming, bipartite graph matching, generalized assignment, etc. For economic efficiency, existing research relies on classic pricing mechanisms such as first-price, second-price, Vickrey-Clarke-Grove (VCG), and critical

value-based Myopic auction-based pricing rules. In Table 1.1, I present a qualitative comparison of the existing studies on auction-based MEC resource allocation mechanisms, comparing their research objectives, QoS criteria, auction framework, and solution approaches.

Existing researches study several approaches of resource allocation mechanism design that maximize offloading task completion rates, yet there exists a significant gap in allocation pricing strategies to optimize the network economics. The heterogeneity in computing resources and wireless resources in a MEC offloading system is often addressed based on: joint allocation of computing and wireless resources [27, 49], multi-dimensional computing resources [29, 46, 60], virtualized instances of computing resources [14], or generalized computing resource units. However, it is unclear how the MEC system model faces heterogeneity in application processors satisfying the task or service-specific QoS criteria. Moreover, the homogeneous pricing strategies determine the allocation prices based on the UE-server pairs. Such matching criteria restrict each MEC server to at most one task offloaded by UE during an offloading period. In a real scenario, however, a single UE may offload multiple tasks with different computing resource requirements. For example, an online gaming UE may simultaneously require computation offloading for the gaming application, VR application, and location-aware navigation application. In such a scenario, to meet the UE's QoS requirements, the offloading tasks should be matched with suitable servers to be processed immediately. Therefore, a heterogeneous resource valuation model implements the flexibility to provision task-specific computing resources and handle multiple offloading re-

Table 1.1: Qualitative comparison of the state-of-the-art

Ref.	Objective	Users' QoS[1]	Resources[2]	Auction Model	Payment Rule	Matching Model[3]	Allocation Algorithm	Economic Properties[4]		
								IC	IR	CE
[51]	Maximize no. of matched pairs	No	CR	Double auction	Critical Payment	many-to-one	Breakeven and dynamic pricing-based heuristic	✓	✓	✓
[58]	Maximize sum profit of MEC clouds	Yes	CR	Multi-round auction	Vickrey auction	many-to-one	Bid performance ratio-based heuristic	-	-	-
[16]	Maximize sum profit of MEC servers	Yes	CR	Online position auction	GSP auction	many-to-one	GSP-based heuristic	-	✓	✓
[28]	Sum utility of servers and UEs	No	CR	Double auction	winning bid	one-to-one	Experience-weighted attraction based heuristic	-	-	-
[25]	Maximize sum utility of MEC clouds and MNO	No	Bandwidth	Randomized auction	Fractional VCG	many-to-one	Greedy heuristic	✓	✓	✓
[29]	Maximize long-term social welfare	Yes	N-types CR	Online double auction	Critical payment	many-to-one	Matching probability-based heuristic	✓	✓	✓
[46]	Maximize sum utility of UEs and SP	No	N-types CR	Forward auction	VCG auction	one-to-many	Dynamic programming-based heuristic	✓	-	-
[14]	Maximize social welfare	Yes	VM	Forward auction	Critical payment	many-to-one	Heuristic with $(\gamma + 1)$ approx. ratio	✓	✓	✓
[59]	Maximize sum profit of MEC clouds	Yes	CR	Online multi-round auction	Vickrey auction	many-to-one	Bid performance ratio-based heuristic	✓	✓	✓
[27]	Maximize expected utility of macro BS	No	CR & WR	second price auction	second price auction	one-to-one	optimal with symmetric Bayesian Nash equilibrium	✓	-	-
[64]	Maximize sum valuation of QoS	Yes	CR	multi-round auction	VCG auction	many-to-one	Kuhn-Munkras algorithm for bipartite graph	✓	✓	✓
[17]	Maximize social welfare	No	CR	double auction	winning bid	many-to-many	heuristic based on minimum cost flow model	✓	✓	✓
[60]	Maximize profits of MEC nodes	No	N-types CR	reverse auction	first price	many-to-one	heuristic based on expected utility theory	✓	✓	✓
[26]	Maximize profit of MEC SP	No	CR	Myerson auction	virtual bid payment	one-to-one	second price	✓	✓	✓
[49]	Maximize social welfare	Yes	CR & WR	Combinatorial auction	Critical payment	many-to-one	Combination of optimal and heuristic	✓	✓	✓

[1] Users' QoS requirements in terms of service latency and energy consumption constraints.
[2] Resources that sellers provides include computing resources (CR), virtual machine (VM) instances, wireless channel resources (WR), and bandwidth
[2] Matching model represents the maximum number of UE (i.e. buyer) that is assigned to a single MEC server/cloud (i.e. seller): *one-to-one* model means one UE can be served at one MEC server/cloud, *one-to-many* model means one UE can be served by more than one MEC servers/clouds at the same time, and *many-to-one* model means multiple UEs can be served at the same MEC server/cloud simultaneously.
[3] Essential economic properties in the design of auction mechanism: incentive compatibility (IC), individual rational (IR), and computationally efficient (CE).

quests from the same UE at task-specific offloading service prices.

Besides, existing works consider heterogeneous MEC servers with different computing capacities under a single MEC cloud (i.e., the seller) or multiple MEC clouds. A large body of research allows for competition among the UEs by submitting bids to win the computing resources from the MEC servers. The existing studies do not investigate the competition among the MEC clouds in the presence of multiple SPs or MNOs. In such a competitive scenario, a MEC cloud behaves strategically to sell as much of its underutilized resources, by choosing an asking price that can attract more UEs to offload computing tasks. Hence, the pricing strategy should ensure stable outcomes so that the bidders cannot degrade others' utility by manipulation. Some existing works consider the double auction model [51, 28, 29, 17] and reverse auction model [60], where MEC servers submit bids expressing their preferences; nevertheless, only a few investigate the convergence and strategic stability of resource allocation pricing policies. For example, [28] proposes a reinforcement learning-based double auction mechanism that allows service providers and users to learn their bidding policies without prior information with homogeneous users' QoS constraints and MEC systems capabilities. Reference [16] proposes adaptive best-response bidding strategies that guarantee computationally efficient resource allocation and locally envy-free equilibrium prices. In [60], the authors develop a federated learning-based reverse auction mechanism, where MEC servers submit their bid preferences along with their respective resource quality scores, and the auction winners are identified according to the sorted scores. However, the main objective of this

work is to encourage high-quality MEC servers to participate in a collaborative learning process.

Furthermore, existing studies do not investigate resource allocation and pricing strategies in a dynamic environment. Reference [65] elaborates on the online auction-based dynamic computation offloading strategies under task deadline constraints and resource allocation strategies under the capacity constraints of the cloudlets. In [14], the authors propose a virtual machine (VM) allocation approach with dynamic transmission delays in a wireless environment; Nonetheless, they do not investigate the dynamics in the processing servers, whereas the computation offloading performance greatly depends on the current computational workloads of allocated VM resources. Therefore, it is imperative to design a service-oriented MEC offloading framework which can be implemented in practice, and guarantees that participating resource sellers are buyers are well off with the resource allocation and pricing strategies for the computation offloading services.

1.4 Thesis Contribution

This thesis introduces the Generalized Second Price (GSP)-based position auction to model the dynamic computation offloading mechanism for MEC-enabled wireless networks. Furthermore, this thesis investigates repeated GSP auction mechanism design to address the resource allocation problem in a dynamic MEC offloading service provisioning framework. As one of the main contributions, this thesis presents a profit-maximizing repeated auction model that determines efficient resource allocation and pricing decisions using GSP-based allocation algo-

rithm. Furthermore, this thesis studies adaptive best-response bidding strategies that ensures efficient allocation prices and satisfies economic properties of auction mechanism design.

The key contributions of this thesis are as listed follows:

- I design a service-oriented multi-user multi-vendor MEC system architecture, specially outlined for deploying the computing offloading as a service trading between wireless UEs and MEC servers.

- I present a basic GSP-based MEC offloading resource allocation mechanism, applying the concepts of position auction model. Considering the proposed auction as a static game of full information, I describe the GSP allocation and pricing rules to determine offloading task assignment and offloading service pricing decisions for MEC offloading.

- I investigate a profit-maximizing repeated GSP mechanism to implement MEC offloading service provisioning via auction in a dynamic wireless network environment. Considering such an online auction as a dynamic game of incomplete information, I design a dynamic resource allocation and pricing mechanism where bidders get the opportunity to maximize their utilities by adapting their bids in every auction round.

- In addition, I study bidding behavior of the competitive MEC resource sellers, and analyze several best-response balanced bidding strategies by applying them on the proposed repeated GSP auction model. I present a novel restricted balanced bidding (RBB) strategy that ensures stable auction outcomes in a dynamic envi-

ronment.

- In order to address the network dynamics in the MEC offloading system, I analyze the dynamic arrival of offloading requests and time-varying computational workloads of the servers using the features of dynamic queuing and priority queuing. I investigate the auction efficiency in terms of satisfying users' QoE and servers' profit gain, by introducing workload-aware dynamic resource management policies in the repeated GSP-based MEC offloading auction mechanism. The main objective of this dynamic auction is to maximize the total valuation of the allocated resources. Furthermore, I derive a novel balanced bidding strategy for the resource sellers that guarantees a symmetric Nash Equilibrium (SNE) resource allocation solution for every auction round. The proposed resource allocation and pricing algorithm for the MEC offloading auction mechanism is computationally efficient and satisfies the economic property of individual rationality.

- Besides theoretical analysis, we perform intensive numerical experiments to validate the proposed repeated auction mechanisms and compare the performance of the proposed GSP mechanism with existing mechanisms.

1.5 Organization of the Thesis

The remainder of the thesis is organized into four chapters as follows

- **Chapter 2** discusses the basic concepts GSP auction model, and the approach to design optimal GSP-based resource allocation al-

gorithm in the context of MEC offloading service provisioning

- **Chapter 3** presents a profit-maximizing repeated GSP auction model for MEC offloading, and studies the bidding strategies of the resource sellers. Detailed MEC offloading system model, auction efficiency analysis, and numerical results are presented.

- **Chapter 3** presents a computational workload-aware repeated GSP auction model for MEC offloading, addressing the heterogeneity in QoS requirements in dynamic offloading scenario. A detailed communication and offloading workflow is presented, along with the utility-maximizing adaptive bidding strategies for resource sellers. The chapter also presents analysis on auction economic properties and equilibrium solution concepts for the proposed resource allocation mechanism. Related system model assumptions and numerical results are given.

- **Chapter 4** provides a summary of the research presented in this thesis, along with discussion future directions, and potential applications of proposed auction-based MEC offloading mechanism in wireless networks.

Symbols and notations used throughout the chapters are given in a table at the beginning of each chapter.

Chapter 2

Repeated GSP Auction Model for MEC Offloading

The GSP mechanism,also known as, position auction has been introduced as keyword auction in the online advertising industry. GSP has been proven to be a successful business model and has been commercially used by the major search engines (e.g. Google, yahoo) for online advertising. In the sponsored search industry, Google introduced the GSP auction to handle the stability issues of the generalized first price auction mechanism (GFP). Over the years, GSP is now considered as the gold standard for revenue maximizing mechanisms in the sponsored search market [40]. Inspired by the practical success of GSP, I investigate the theoretical properties of GSP to design new auction mechanisms that enhance MEC offloading system performance and also provide incentives to the service providers.

In this chapter, I first discuss the basics of designing a generalized second price (GSP) auction model, along with allocation and pricing rules, and optimal GSP auction concepts. Then, the equilibrium solution concepts for GSP mechanism and the idea of optimal GSP auction design are described.

and present a simplistic GSP auction-based resource allocation and pricing framework for MEC offloading. Next, I discuss the prospects of GSP-based repeated auction mechanisms to deploy MEC services and efficiently manage the computing resources at the MEC servers.

2.1 GSP Auction Mechanism Design

This section introduces the basic features of a generalized second price (GSP) mechanism and various equilibrium concepts of GSP auctions. Next, I present a GSP auction-based resource allocation and pricing framework for MEC offloading.

2.1.1 Basic Features of GSP Auction

The standard GSP mechanism for keyword auctions, consists of the following features [54, 12]:

- A set of k advertising slots, i.e., the positions on the web page.
  - Each slot is associated with a click-through-rate (CTR) $\alpha_{i,s} = q_i\theta_s$, which represents the rate of getting clicks from user on the ad of bidder i when it is placed at position s. Here, q_i is bidder-specific term that denotes that a user will click on the ad of bidder i , and the position-specific term θ_s denotes the probability that a user clicks on the ad in slot s.
  - The probability of receiving clicks in the higher position is higher than the position at the bottom, i.e., $\theta_1 > \theta_2 > \cdots > \theta_k$.
- A set of n advertisers, who want to sell some sort of products or services and thus participate in the ad auction to advertise their

products.

  - Typically, it is assumed that $n > k$ because large number of bidders compete for a limited number of ad slots in most of the practical scenarios.
  - Each advertiser has its own private valuation $v_i > 0$ for a user click.
  - An advertiser has preference over higher slot as it has higher probability to get more user clicks.
  - Each advertiser i chooses a set of keywords related the product and determines a bid b_i with for each keyword that basically represents the amount it is willing to pay if a user clicks on its ad.
  - The bid b_i does not necessarily equal to the value v_i due to the strategic behavior of the bidders. The bid vector or bidders' strategic profile is mathematically represented as $\mathbf{b} = (b_1, \ldots, b_n)$.

- An auctioneer or search engine, that determines the allocation and pricing decisions based on mathematical structure similar to existing two-sided matching models [42].

In the theoretical studies, the basic setting of the GSP mechanism is considered as a static one-shot game with complete information, where the bidders have full knowledge of the advertising slots and corresponding CTRs. Furthermore, it is assumed that the bidders have no budget constraint, and submit a single-dimensional bid based on the CTR information [40].

Table 2.1: List of key notations used in Chapter 2

Notation	Description
k	Number of advertising slots
n	Number of advertisers/bidders
v_i	Private valuation/user-click of advertiser i
b_i	Amount that bidder i offers to pay for each user-click
θ_s	Rate of user clicks on the s-th advertisement slot
q_i	Probability that a user clicks on the bidder i's advertisement
$\alpha_{i,s}$	Rate of user clicks on bidder i' advertisement in the s-th slot
r_i	Ranking score of bidder i
π_s	Index of the advertiser assigned to the s-th slot
p_s	Allocation price for the s-th slot
$u_{i,s}$	Utility gain of bidder i from slot s
N	Number of MEC applications
I	Number of MEC servers
K_n	Number of offloading tasks of n-th application
λ_k	Computing resource consumption rate of k-th task position
d_k	Offloading data size of k-th task
L_n	Computation processing density of n-th application
w_i	Number of vCPUs at i-th server
u_i	Utility of i-th server

2.1.2 Allocation and Pricing Rules of GSP Auction

The GSP mechanism has mainly two fundamental principles [40]:

- **Ranking rule**: The ranking rule basically helps determine the allocation of ad slots to the bidders. The ranking is determined by a ranking function $r_i(q_i, b_i)$ that computes a score for ranking based on the CTR value of the bidder's ad and the bid value.

There are mainly two ranking approaches:

- *Rank-by-Bid*: Bidders are ranked in descending order of their bids with the ranking score $r_i(q_i, b_i) = b_i$.
- *Rank-by-Revenue*: The ranking is determined as descending order of the ranking score $r_i(q_i, b_i) = q_i b_i$.

In the literature, there are other ranking approaches to model the ranking function based on CTR or even without CTR [1, 24, 53, 41].

The allocation of k slots is done according to the ranking scores. The bidder $i \leq k$ is matched to slot in position i, where the slot with the highest CTR to the bidder with highest bid, the second best slot to the second highest bidder, and so on. I denote the allocation $\pi_s = i$ which represents the identity of the advertiser i that is assigned to slot s.

- **Pricing rule**: The pricing rule determines the price that each bidder pays to the search engine when user clicks on its ad. The pricing rule depends on the ranking approach that has been taken to determine the allocation.

  - In the *Rank-by-Bid* approach, the allocation price is equal to the bid submitted by the bidder ranked next to him, i.e., $p_s = b_{i+1}$.
  - In the *Rank-by-Revenue* approach, the allocation price is computed as,

$$p_s = \frac{q_{i+1} b_{i+1}}{q_i}.$$

The net profit or utility that an advertiser i can expect to gain from winning the slot s is given by, $u_{i,s} = (v_i - p_s)\theta_s$.

The standard GSP mechanism can be defined as follows:

Definition 1 *The generalized second price (GSP) mechanism for keyword auctions determines the allocation and pricing decisions as follows:*

- *Bidders are allocated slots in decreasing order of their ranking score r_i.*
- *For each slot s, the payment p_s of bidder π_s is determined based on bid b_{i+1} of the bidder ranked next to winner i.*

Bidders who do not win a slot make no payment and and gain no utility.

2.2 Repeated GSP Auction and Equilibrium Solutions

The GSP or position auctions are generally repeated over discrete time slots for applications in real world scenario. This repeated GSP auction mechanism can be considered as a dynamic game of incomplete information when implemented in wireless network environment. Alternatively, a single round of GSP auction can be thought as a static game of complete information, which is then repeated at different time intervals avoiding the complex multi-period information sets. Although the auction is repeated with same set of allocation and pricing rules, the auction may result into different sets of outcomes depending on the current information set. The different auction outcomes in different

rounds or time slots are referred to as the states in a set of equilibria for the auction game.

The following section briefly introduces some of the equilibrium concepts used in literature to study the repeated GSP auction mechanism under both full-information and incomplete-information settings. Next, I discuss different approaches to design an optimal GSP auction, satisfying the equilibrium conditions in the dynamic environment.

2.2.1 Equilibrium Concepts in GSP Auction

In game theory, a Nash equilibrium [36] is considered as the stable outcome of a game, that is, a situation where no player can improve his or her payoff (utility) by a unilateral strategy change. The Nash equilibrium in an auction framework holds the following inequality:

$$u_i(b_i, b_{-i}) \geq u_i(b_{i'}, b_{-i}), \forall b_{i'}, \tag{2.1}$$

where u_i is the utility of bidder i, b_{-i} represents the bids of all other bidders, and $b_{i'}$ is alternative bid of bidder i.

In case of GSP auction game, there always exists a Nash equilibrium in the complete-information setting, when no bidder would have an incentive to obtain a different slot other than the currently assigned one [12, 54]. When a bidder changes the bid in order to change his or her utility it also changes his or her slot. Thus, the Nash equilibrium in the GSP auction can be defined as follows:

Definition 2 *A set of bids* $\mathbf{b}$ *is a* ***Nash Equilibrium****, if for every*

slot s the allocation of every bidder satisfies the following inequalities:

$$\theta_s (v_s - p_s) \geq \theta_j (v_s - p_j), \quad \forall j > s, \tag{2.2}$$

$$\theta_s (v_s - p_s) \geq \theta_j (v_s - p_{j-1}), \quad \forall j < s, \tag{2.3}$$

In GSP auction under complete-information setting, the notion of the symmetric Nash equilibrium (SNE), is the stable outcome when every bidder prefers to purchase the slot it is currently in rather than some other slot. So, there should be no incentives for any pair of bidders to swap their slots. The SNE is a subset of Nash equilibria and achieves the maximal revenue among all Nash equilibria [54]. The definition of SNE is as follows:

Definition 3 *A set of bids* **b** *is a* ***Symmetric Nash Equilibrium (SNE)****, if satisfies the following inequalities:*

$$\theta_s (v_s - p_s) \geq \theta_j (v_s - p_j), \quad \forall j, s, \tag{2.4}$$

Another popular notion of GSP equilibrium in the complete information game setting, is the locally envy-free equilibrium (LEFE), which is also a subset of Nash equilibria and has an equivalent concept as SNE [12]. The definition of LEFE is given as follows:

Definition 4 *A set of bids* **b** *is a* ***Locally Envy-Free Equilibrium (LEFE)****, if no bidder can improve his or her utility by exchanging bids with the bidder ranked one position above him or her:*

$$\theta_s (v_s - p_s) \geq \theta_{s-1} (v_s - p_{s-1}), \quad \forall s, \tag{2.5}$$

2.2.2 Optimal Repeated GSP Auction Design

When a static game is repeated over an extended period of time in the dynamic environment, the players get the opportunity to learn many characteristics about the game. In case of repeated GSP auction mechanism, bidders can learn about the allocation and pricing rules based on the auction outcomes from the previous round. So, they can adjust their bids accordingly, intending to maximize their utilities. However, the main challenge lies in identifying the optimal equilibrium, which yields maximum utility gain for each participating players. When the repeated GSP auction reaches the optimal equilibrium point, it is desired that no player deviates from the current state by overbidding/underbidding. The criteria of selecting the optimal equlibirum is to find the most relevant approximation of the equilibrium of the dynamic game, out of a set of equilibria from the static games of complete information. The idea of designing optimal repeated GSP auction is to determine an upper bound on the allocation valuation from the equilibria of the static games, and then excluding the equilibrium of the dynamic setting that exceeds the upper bound [13].

Besides, in an optimal auction it is desired that every bidder truthfully reports their actual valuation as bids. However, repeated GSP auction is designed to enable strategic interactions among the players. The bidders in the GSP auction do not necessarily behave truthfully; rather they bid strategically to maximize their bids [12]. Therefore, it becomes more challenging to identify optimal equilibrium in GSP auction, that efficiently regulates the strategic interactions among the players. One of the effective approaches to enforce such regulations,

is to design the pricing mechanism in GSP auction with some reserve prices. The analysis of reserve prices on the GSP auction reveals direct and indirect impacts on bidders' bidding behavior. In the repeated GSP auction model, the optimal reserve price is independent of the CTR rate. Moreover, As long as there are more positions than the number of bidders, the reserve price directly affects the lowest bidder and the lower bidder's payment becomes equal to the reserve price [13]. This implies the impact of reserve prices on the equilibrium behavior of the bidders. When reserve price increases, the total payment of every bidder who wins a slot (except the last winning bidder) increases by the same amount. The optimal reserve price also affects the total surplus and auctioneer's revenue.

The GSP auction has successfully been applied in commercial search engines for online advertising. There still exist quite a number of open research problems on GSP auctions, which include but not limited to: approximation of optimal efficiency in online auctions, estimation of CTRs considering stochastic factors of the dynamic environment, mechanism design with certain revenue guarantee, learning bidders' behavior considering externalities, optimal allocation mechanism with budget constraints [40].

2.2.3 Repeated GSP Auction Model for MEC Offloading

The simple mathematical structure, stable assignment outcomes of the two-sided matching game, and economic properties GSP auction make it a promising choice to implement dynamic resource provisioning in the MEC platform. In this section, a reverse GSP auction is modeled demonstrating how MEC offloading can be implemented using the GSP-

based position auction framework. This section mainly focuses on how the resource allocation and pricing mechanisms for MEC offloading can be modeled using the two-sided matching game structure of the GSP-based keyword auction. Then, I describe the winner determination problem formulation and discuss solution approaches to determine the allocation and pricing rules. A walk through example, illustrating the resource allocation and pricing procedures, is also given.

The GSP auction has certain characteristics that allows MEC platform to efficiently manage the heterogeneity of users' offloading resource demands and also allow computing service providers to determine their pricing strategies in an competitive environment. The MEC platform accommodates the features of GSP auction by creating N different application slots that can be considered as the keywords from the GSP auction. The application slots are placed on top of the virtualization infrastructure to compute the users' offloaded data using the computing resources from the service providers. The SDN controller matches the users' offloading requests with application criteria and then prepares the virtualization infrastructure manager to process the tasks. The SDN controller also ensures the integrity and authenticity of offloading requests, application processing rules and requirements, and if necessary adjusts the policies to comply with the latency and resource availability constraints. Now, assume there are K_n offloading requests for the mobile edge application in the slot n and thus MEC platform manager defines at least K_n virtual machines (VMs) and maps them to the corresponding tasks in the application slot.

The VMs associated with offloading tasks can be considered as the

resource consumers, that utilizes CPU, memory, power, storage, and network resources of the MEC servers. The MEC virtualization infrastructure manager check whether there is enough available resources before activating the VMs, and then compartmentalizes a resources which is basically a logical abstraction of the resources available at MEC servers across different service providers. The VMs are usually defined with share-based percentage of the total CPU, memory, and storage I/O [56]. For simplicity, I consider the virtual CPUs as the smallest unit of resource partitions allocated to VMs. A VM can be allocated more than one virtual CPUs depending on the user's offloading application requirements.

I now present a GSP-based MEC offloading system model, where the SDN controller acts as the auctioneer who manages the overall functionalities of offloading process starting from receiving users' requests to determining the resource allocation and pricing decisions. The set of K auction slots in the proposed GSP-based auction mechanism, represents the offloading tasks in the MEC system. The offloading tasks with the highest resource consumption rate is placed at the top position, the task with the second highest resource consumption rate is located at the second position, and so on. Thus, I consider the offloading tasks arranged in K hierarchical positions with distinct resource consumption rates, e.g., $\lambda_1 > \ldots, > \lambda_K$. Each user can request at most one type of application during an offloading period. However, the same application may be requested by more than one user and the tasks are scheduled according to their resource consumption rates.

I consider a set of offloading tasks $\mathcal{K} = \{\mathcal{K}_1\} \cup \cdots \cup \{\mathcal{K}_N\}$ consisting

of all offloading requests over N applications. I assume that the MEC platform prioritizes the tasks based on the amount of CPU resources actively used by the VMs assigned to the offloading tasks. Hence, I define the parameter *resource consumption rate*

$$\lambda_k = d_k L_n \tag{2.6}$$

where d_k is the data size of the offloading task (in bits) and L_n is the application-specific computation processing density (in CPU cycles/bit) required to run the application of type n. The offloading task with the higher resource consumption rate has higher resource demands which also increases the probability of getting higher profits from resource provisioning.

I consider a set of I MEC servers owned by different computing service providers, who participate into the auction to sell their computing resources to offloading users. Let w_i be the amount of virtual CPUs available at MEC server i and $v_i > 0$ be the server's private valuation of each virtual CPU. In the auction mechanism, the MEC servers compete with each other to win the offloading tasks and bids rationally with the objective of maximizing their own profits.The bid b_i represents the minimum amount the MEC server i wishes to receive for the allocation of each virtual CPU. Therefore, the bid value does not necessarily equal to the value v_i. Each MEC server has preference over offloading tasks in the higher positions due to the probability of receiving higher payments.

2.2.4 GSP-Based Resource Allocation Algorithm Design

In the GSP-based MEC offloading auction model discussed above, the SDN controller takes into the bids submitted by MEC servers and the offloading requests from users, and then matches the offloading tasks to suitable MEC servers. Following are the ranking and allocation rules to assign the offloading task slots to the bidders:

- Considering the *rank-by-revenue* approach, I define the ranking score for MEC server as $r_i = w_i/b_i$, such that bidders with lower bid and higher resource availability get higher ranking positions.

- The proposed mechanism then assigns the K offloading task slots according to the descending order of ranking scores, such that bidder with the highest ranking score gets the offloading task with the highest resource consumption rate.

  I consider the simple one-to-one matching scenario as the basic GSP auction model, where one offloading task is assigned to at most one MEC server, and vice versa. If there is less number of tasks than the MEC servers then a dummy task slot is considered with a data size equal to zero. However, it is more likely that a MEC server need to handle multiple offloading tasks simultaneously in the practical scenario, which I study in the following chapter.

- I denote the assignment of the offloading task in slot k to the MEC server i as $\pi_k = i$.

- Next, the mechanism partitions and allocates the computing resources from the winner MEC server i into the VM that computes

the assigned offloading task in slot k. The amount of CPU resources allocated to each task assignment is computed as follows:

$$x_{i,k} = \left\lceil \min \left\{ \frac{\lambda_k}{c_i}, w_i \right\} \right\rceil, \tag{2.7}$$

where c_i is the computing processing capacity of each CPU from MEC server i.

- The resource allocation price is determined based on his or her resource availability and the ranking score of the next bidder in the rank as follows,

$$p_k = \begin{cases} w_i/r_{i+1}, & \text{if } 1 \leq k \leq K, rank(i) < I, \\ w_i/r_i + \epsilon, & \text{otherwise} \end{cases} \tag{2.8}$$

where ϵ a small positive constant.

- The utility of each bidder is thus given by

$$u_i = (p_k - v_i)x_{i,k} \tag{2.9}$$

Example 2.1 ***A walk through example on how GSP-based auction outcomes are computed for MEC resource allocation and pricing decisions.***

To better understand the GSP mechanism for the MEC offloading system, consider $K = 3$ users request for offloading services on $N = 3$ different applications. I represent the set of users' offloading tasks as $\mathcal{K} = \{k_1, k_2, k_3, k_4\}$, edge applications as $\mathcal{N} = \{n_1, n_2\}$, and set of bidders' as $\mathcal{I} = \{i_1, i_2, i_3, i_4\}$. I assume that users k_1 and k_2 request for the application n_1 and the other user k_3 request application n_2. Consider, the offloading data sizes are given as $\mathbf{d_k} = \{6, 5, 3, 2\}$, and

	k	d_k	λ_k	π_k	r_i	v_i	w_i	c_i	$x_{i,j}$	p_k	u_i
N_1	k_1	6	60	i_2	20	2	40	3	20	2.7	14
	k_2	5	50	i_3	15	1	15	2.5	20	1.1	2
N_2	k_3	3	15	i_4	14	2	28	3	5	2.8	4
	k_4	2	10	i_1	10	3	30	2	5	3.1	0.5

Figure 2.1: An example demonstrating GSP allocation and pricing

applications' computation processing densities are $L_{n_1} = 10$ and $L_{n_2} = 5$. Therefore, the resource consumption rates are computed as $\lambda_{\mathbf{k}} = \{60, 50, 15, 10\}$ as shown in Fig. 2.1.

Assume, the bids and available resource units of MEC servers are $\mathbf{b_i} = \{3, 2, 1, 2\}$ and $\mathbf{w_i} = \{30, 40, 15, 28\}$ respectively. The ranking scores can be computed as $\mathbf{r_i} = \{30/3, 40/2, 15/1, 28/2\} = \{10, 20, 15, 14\}$. Then, I rank the bidders according to their ranking scores as $r_2 > r_3 > r_4 > r_1$ and assign the tasks starting from matching the top task slot to the bidder with the highest slot (See Fig. 2.1). Considering the CPU computing capacity $\mathbf{c_i} = \{2, 3, 2.5, 3\}$ (GHz), I compute the amount of allocated resource units and the allocation prices using eqn. (2.7) and eqn. (2.8) respectively. For an instance, task k_1 is assigned to MEC server i_2, with the decision variables as $\pi_{k_1} = i_2$, $x_{i_2,k_1} = 60/3 = 20$, and $p_{k_2} = \frac{40}{15} = 2.7$. Therefore, the utility of MEC server i_2 becomes, $u_{i_2} = (2.7 - 2)\,20 = 14$.

2.3 Conclusion

In this chapter, the basics of GSP auction model is described along with the characteristics of designing repeated GSP mechanisms. The notions of the optimal GSP auction and corresponding equilibrium so-

lution concepts are also presented. Then, a simple MEC offloading framework is presented applying the GSP mechanism design concepts, followed by an walk-through example on how to obtain GSP-based resource allocation and pricing decisions for computation offloading. In the next chapter, a detailed repeated GSP mechanism is modeled for MEC offloading that maximizes the profits of the computing service providers.

Chapter 3

Profit-Maximizing Repeated GSP Auction Model for MEC Offloading

Addressing the heterogeneity of users' demands for computation offloading in a multi-vendor MEC network, I investigate the resource allocation problem in this chapter, with the goal of maximizing the profits of MEC servers. This chapter presents a reverse GSP auction framework based on *position auction*, addressing the winner determination, resource allocation pricing, and bidding strategy optimization problems for deploying computation offloading in MEC networks.

An overview of the MEC offloading resource allocation problem and related challenges are discussed in Section 3.1. The MEC offloading system model and related assumptions to design the GSP-based resource allocation mechanism are outlined in Section 3.2. Next, I present the proposed GSP-based reverse position auction, along with the formulation and solution approach to the winnper determination problem (WDP) in Section 3.3. In Section 3.4, the analysis on the auction efficiency is given. Numerical results in Section 3.5 demonstrates the performance of the proposed auction mechanism, and finally Section 3.6 concludes the chapter.

3.1 Introduction

One of the fundamental challenges of mobile edge computing (MEC), which allows the mobile users to offload their computationally-intensive tasks to the servers located at the network's edge, is to develop methods to efficiently allocate the limited computational resources of the edge servers to the offloading users. To address this challenge, existing researches study designing efficient computation offloading scheduling and resource allocation methods, focusing on optimizing the performance metrics from the users' perspectives.

Considering the heterogeneity of users' demand in a multi-MEC server scenario, Zhang *et al.* [61] propose a combinatorial auction model to study a matching problem, where an MEC server is assigned to the users with allocation of the requested bundle of resources. In such an auction framework, the crucial job of the bidders (i.e. users) is to evaluate the computing resources across MEC servers and to decide which server they prefer to offload their tasks to. Bahreini *et al.* [4] combine the features from position auction with combinatorial auction to model the users' preferences over the computing resources at the edge and cloud servers. They propose a bidding preference model for users that ensures envy-free allocation, i.e. no user can improve her utility by exchanging the allocation with other users. The online resource auction mechanism in [65] allows the users to dynamically evaluate the computing resources at the edge servers that maximizes their long-term utility. However, the following question remains open: How the users can characterize the true evaluation of the optimal bid such that the auction does not result in unfair allocation prices?

From the economical point of view, an efficient resource allocation or trading mechanism ensures non-negative profit for the participating edge providers. Abbas *et. al.* [22] consider a profit maximization problem for the service providers with limited resources. Considering, users as the bidders, the authors propose a two time-scale auction-based approach to solve the resource allocation and pricing problem. The edge providers can also be modeled as strategic players alongside the offloading users in a double auction framework. Jin *et. al.* [19] propose a double auction mechanism, where the auctioneer matches mobile users with cloudlets for trading computing resources. The outcome of such a procedure is a one-to-one matching between the cloudlet and the user, which is not realistic for practical implementation. This issue has been addressed in [51], where the double-auction mechanism allows an edge server to serve more than one user at a time; however, it ignores the heterogeneity of offloading requests for different tasks/application.

In order to address the shortcomings of the current literature as discussed above and I consider a novel approach to develop a reverse auction model based on the GSP-based position auction, solving the resource allocation problem. The main goal is to efficiently allocate the computational resources of the edge servers to the offloading users with different computational capacity requirements. The allocation should guarantee the users' QoE and provide both the users and the providers with positive economical gain, which corresponds to lower service cost for users and higher efficiency in the resource utilization for servers. In addition, each edge server can serve multiple users at the same time.

3.1.1 Research Contribution

The main contributions of this chapter are summarized as follows:

- I model a reverse auction framework for MEC offloading using the features of position auction and generalized second price (GSP) pricing rules.
- I propose a greedy bidding strategy for the MEC servers to compute their bids maximizing their utilities.
- I formulate a combinatorial optimization problem as the winner determination problem (WDP) for the proposed auction model and design an approximation algorithm to solve the problem in polynomial time.
- I provide theoretical and numerical analysis on economic and computational performance for the proposed auction mechanism.

3.2 System Model and Assumptions

I consider an MEC offloading system where a software-defined network (SDN) controller acts as a broker between the offloading users and computation service providers (CSPs). I denote the set of offloading users and computation service providers (CSPs) as $\mathcal{K} = \{1, \ldots, K\}$ and $\mathcal{I} = \{1, \ldots, I\}$, respectively. Each CSP has a single multi-core MEC server and I use index $i \in \mathcal{I}$ to denote a CSP and an MEC server interchangeably. Users in different cells use orthogonal radio channels to transmit to their corresponding base stations and thus to the SDN controller in the core network. The MEC servers of the CSPs can com-

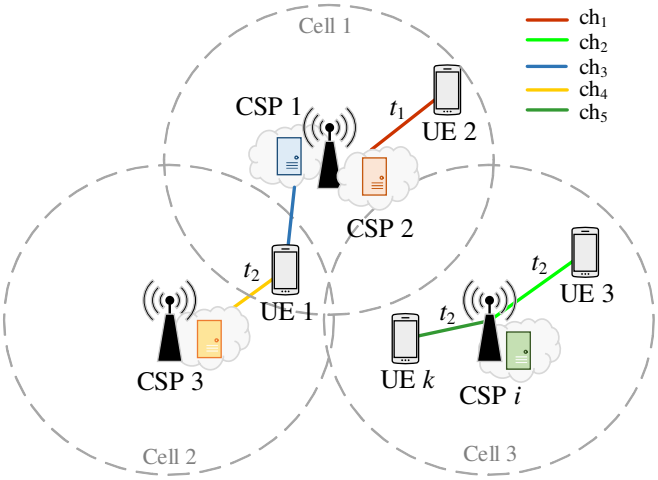


Figure 3.1: An MEC offloading scenario.

municate with the BSs through the core network. The communication delay in the core network is assumed to be negligible.

The offloading mechanism operates in discrete time slots of equal offloading period Δt. At the beginning of each offloading period, the SDN controller receives the offloading requests. The task of SDN is to schedule each offloading request to a suitable MEC server and also to efficiently allocate the computational resources. To perform this task, the SDN runs a reverse auction to get the computing resources from MEC servers as much as required to satisfy users' demands and allocate them for computing the offloading tasks. The resource allocation is performed depending on the arrival of offloading requests in each time slots. The system remains idle if there no offloading request during a time slot. Moreover, when all the MEC servers are busy and there is no resource available, then the incoming offloading requests will remain unassigned[1] and will eventually be rejected at the end of the time slot.

[1]The unassigned tasks can be buffered in a queue so they can be computes in the next time slot. This will change the dynamics of resource allocation mechanism and therefore I left this as future work.

I describe the procedure in detail throughout the chapter. The symbols and notations used in this chapter is listed in Table 3.1.

3.2.1 MEC Offloading System Model

Users can request for N different types of applications of tasks for MEC offloading, gathered in a set $\mathcal{N}$. The task type $n \in \mathcal{N}$ is identified by its computation processing density requirement, denoted by L_n in CPU cycles/bit. I represent the popularity of the task types by its *hit ratio*[2], $\theta_1 \geq \theta_2 \geq \cdots \geq \theta_N$, where $0 \leq \theta_n \leq 1$.

A user's offloading request is represented by the tuple $\langle d^k, g^k \rangle$, where $d^k \in [0, d^k_{\max}]$ is the offloading data size (in bits) which may vary over different offloading periods and $g^k \in \mathcal{N}$ denotes the requested task type. Each user requests for only a single type of application during each offloading period; however, the same application n can be requested by more than one user.

The CPU consumption rate for k-th task of type n is given by $\lambda^k_n = d^k L_n$. The offloading tasks that require large amount of CPU resources has higher priority over the less computationally-intensive tasks. Consequently, offloading requests for each task type are sorted in decreasing order of their CPU consumption rates, i.e. the amount of CPU cycles utilized during each offloading period.

The offloading transmission rate of a user depends on the wireless channel condition and the spectrum allocation. I assume that the channels' condition remain unchanged during each offloading period. Let B_i be the allocated bandwidth. Also, p^k indicates the transmission power

[2]The hit ratio is the ratio between number of users requesting for a particular app and the total number of requests observed over a period of time.

of user k and h^k is the channel power gain between user k and the corresponding BS. The achievable uplink data rate (in bits/sec) of user k is then given by

$$r_i^{k,\text{up}} = B_i \log_2 \left(1 + \frac{p^k h^k}{\sigma^2}\right), \tag{3.1}$$

where σ^2 is the noise power. Similarly, the downlink transmission rate $r_i^{k,\text{down}}$ can be calculated when the downlink transmit power is being p^i. Thus the uplink and downlink transmission delays yield $\delta_i^{k,\text{up}} = d^k/r_i^{k,\text{up}}$ and $\delta_i^{k,\text{down}} = \hat{d}^k/r_i^{k,\text{down}}$ respectively, where $\hat{d}^k$ is the size of the computation result. The computation delay at server i is $\delta_i^{k,\text{comp}} = \lambda_n^k/c_i$.

The waiting time of user k at the MEC server depends on the computation time of the users who are ahead of user k at the server end. Since the tasks are ranked according to their CPU consumption rate, the waiting delay of user k can be represented as

$$\delta_i^{k,\text{wait}} = \sum_{k'=1}^{\text{rank}_k - 1} \left\{ \delta_i^{k',\text{comp}} | \, \text{r}_{k'} < \text{r}_k, k' \neq k \right\}. \tag{3.2}$$

The total execution delay consists of the uplink transmission delay, waiting delay at the server, computation (processing) time, and the downlink transmission delay. The total execution delay for computing user k's task can be expressed as

$$\delta_i^k = \sum_{i=1}^{I} \left(\delta_i^{k,\text{up}} + \delta_i^{k,\text{wait}} + \delta_i^{k,\text{comp}} + \delta_i^{k,\text{down}} \right) y_{i,n}^k \ , \tag{3.3}$$

where $y_{i,n}^k \in \{0, 1\}$ is a binary decision variable which represents whether CSP i is assigned to compute the task of type n or not.

The resource demand of user k is considered as the amount of CPU resource units required to compute the requested task of type n, which

is given by $a_{i,n}^k = \left\lceil \frac{\lambda_n^k}{c_i} \right\rceil$. Thus, the amount that user k pays for allocation of $a_{i,n}^k$ units of computation offloading is given by

$$\beta_i^k = \sum_{i=1}^{I} \pi_n a_{i,n}^k y_{i,n}^k. \tag{3.4}$$

Naturally, the users' utility functions shall measure the satisfaction level with the computation offloading service taking the total delay and monetary cost into account. Considering $\delta_k^{\max}$ and $\beta_k^{\max}$ as user k's maximum tolerable delay and a monetary budget, respectively, I define the satisfaction function of user k for the computation service received from MEC server i as

$$\phi_i^k = \alpha_k^a a_{i,n}^k - \alpha_k^d \frac{\delta_i^k}{\delta_{\max}^k} - \alpha_k^p \frac{\beta_i^k}{\beta_k^{\max}}, \tag{3.5}$$

where α_k^a, α_k^d, and α_k^p are the scaling parameters. I define the satisfaction index γ_i^k which indicates how much the user k is satisfied with the MEC server i. It is given by [32]

$$\gamma_i^k = 1 - e^{-\theta_n \phi_i^k}. \tag{3.6}$$

The utility of an MEC server for winning tasks of different types depends on her expected profit and related allocation cost. Each MEC server incurs some energy cost for allocating its resources for computing offloading tasks. The energy consumption cost of MEC server i for each resource unit is estimated as $e_i = \alpha_i^e \left(P_0 + \kappa_i c_i^3\right)$, where P_0 is the fixed initial power required for server activation, κ_i is a constant parameter that represents the effective switched capacitance of each core. Here, α_i^e is the scaling parameter to define the monetary cost for each unit of energy consumption. Moreover, the value of each CPU resource unit to

Table 3.1: List of key notations used in Chapter 3

Notation	**Description**
N	Number of MEC applications
K	Number of Offloading users
I	Number of MEC servers
L_n	Computation processing density of n-th application
θ_n	Task popularity or hit-ratio of n-th application
λ_n^k	CPU consumption rate of k-th task of type n
d^k	Offloading data size of k-th task
g^k	Offloading task type
$r_i^{k,\text{up}}$	Uplink data rate of user k
δ_i^k	Total execution delay of k-th task at server i
c_i	Computation power of server i
$a_{i,n}^k$	Number of vCPU resource unit required for k-th from server i
β_i^k	Amount paid by user k to server i for computation offloading
ϕ_i^k	Level of satisfaction of user k with server i
γ_i^k	Satisfaction index of user k for server i
e_i	Energy consumption cost of server i
ρ_i	Private valuation of server i
v_i	Valuation of a vCPU resource unit at server i
b_i	Bid price of server i
w_i	Number of vCPU resource units available at server i
$x_{i,n}$	Allocation decision variable
$y_{i,n}^k$	Amount of vCPUs allocated to k-th task from server i
π_n	Allocation price of task type n
V	Auction welfare
$u_{i,n}$	Utility of server i for task type n
η_i	Effective bid of server i

MEC server i depends on a private value $\rho_i \in [0, \rho_i^{\max}]$ as well as the hit ratio θ_n, which is known. The hit ratio indicates the bidder's preference over different task types based on the task popularity. Therefore, the server's total valuation of each resource unit can be represented as $v_i = \rho_i + e_i$.

Considering that $x_{i,n}$ is the amount of SDN decides to procure from MEC server i to compute tasks of type n and allocation price is equal to π_n, the MEC server's utility function yields

$$u_{i,n} = (\pi_n - \theta_n v_i)\, x_{i,n}, \tag{3.7}$$

3.3 Repeated GSP-Based Reverse Auction Model

In this section, I present a GSP-based reverse auction model for MEC offloading is discussed in details. I first design a position auction in the reverse auction format where the SDN controller acts as the auctioneer and the users participate as buyers who wish to purchase computing capacities to execute their computation-intensive tasks. In this reverse auction model, the MEC servers play the roles of bidders who want to sell their residual computing capacities as virtual instances of CPU resource units and provide computation as a service to offloading user via the SDN controller. Next, I study the bidding strategy of MEC servers' from sellers' perspectives in a reverse auction and formulate the winner determination problem that the auctioneer solves to obtain the auction outcomes. Then, I propose solution approach for solving the optimization problem and discuss the pricing function designed based on generalized second price (GSP) mechanism [54]. Finally, I analyze the computational complexity and economic properties of the proposed

auction mechanism.

I design a position auction where the SDN controller creates $n = 1, \ldots, N$ auction slots/positions for each task type and then captures users' offloading requests $\langle d^k, g^k \rangle$. Upon the arrival of users offloading requests, the SDN controller determines users' resource demands, and then issues *request for proposal* (RFP), looking for suitable CSPs. The RFP specifies the task types, task hit ratio θ_n, and allocation prices $\pi_n^{(t-1)}$ from the previous round of auction.

I define MEC server i has w_i units of CPU resources which they want to sell via auction, and the difference of the auctioned computing capacities is determined by $c_i = m_i f_i$, where m_i is the number of CPU cores and f_i is the CPU-cycle frequency of each core in i-th MEC server. In response to the RFP, an MEC server employs a greedy bidding strategy to determine her bid in a way that maximizes her utility (as defined in the following subsection). Let $b_i = (b_{i,1}, b_{i,2}, \ldots, b_{i,N})$ be the set of ask prices, where $b_{i,n}$ is the price per CPU resource unit for each task type $n \in \mathcal{N}$. The bid submitted by i-th MEC server is thus represented as $\langle b_i, w_i \rangle$.

3.3.1 Bidding Strategy of MEC Servers

I propose an adaptive bidding strategy for MEC servers using the restricted balanced bidding (RBB) approach [9]. I consider $\mathcal{N}^{(t)}$ is the set of task types to be assigned in the current time-slot t and $\mathcal{N}_i^{(t-1)}$ is the set of task types that the MEC server i has won in the $(t-1)$ round. The MEC server i chooses the same bid as previous for the task types $n' \in \mathcal{N}_i$, assuming that she would still win tasks for same the types in the next round as other bidders do not change their bids. In fact, the

MEC server i targets to win task type(s) that she has not won in $(t-1)$ round and thus updates bids only for the task types $n \in \mathcal{N}^{(t)} \backslash \mathcal{N}_i^{(t-1)}$, where $\theta_n \leq \theta_{n'}$.

Given the allocation outcomes from the last auction round, the MEC server i chooses her bid such that marginal bid for each task type is at least equal to the marginal valuation and satisfies

$$\pi_{n-1}^{(t-1)} x_{i,n-1}^{(t-1)} - b_{i,n}^{(t)} x_{i,n}^{(t-1)} \geq \theta_{n-1} x_{i,n-1}^{(t-1)} v_i - \theta_n x_{i,n}^{(t-1)} v_i, \tag{3.8}$$

where $b_{i,n}^{(t)}$ is the bid submitted by MEC server i for task type n in the auction round t. The RBB strategy for the proposed position auction is defined as follows.

Definition 5 *Given the allocation outcomes $(x_{i,n}^{(t-1)}, \pi_n^{(t-1)})$ for all of the announced task types, the Restricted Balanced Bidding (RBB) strategy of the MEC server i is defined as follows:*

(i) Target the task types $n \in \mathcal{N}^{(t)} \backslash \mathcal{N}_i^{(t-1)}$ with lower task popularity $\theta_{n'} \leq \theta_n$ than the ones won in the previous round, and choose the next bid as

$$b_{i,n}^{(t)} = \begin{cases} \theta_n v_i, & \text{if } n = 1 \\ \theta_n v_i + \dfrac{x_{i,n-1}^{(t-1)} + x_{i,n}^{(t-1)}}{2} \left(\pi_{n-1}^{(t-1)} - \theta_{n-1} v_i \right), & \\ & \text{if } 2 \leq n \leq N. \end{cases} \tag{3.9}$$

(ii) The bids for all other task types that bidder i has already won remain same as the previous round, $b_{i,n}^{(t)} = b_{i,n}^{(t-1)}$

3.3.2 Winner Determination Problem Formulation

After receiving the bids $\langle b_i, w_i \rangle$ from MEC servers, the SDN controller decides how to allocate the offered CPU resources that would satisfy

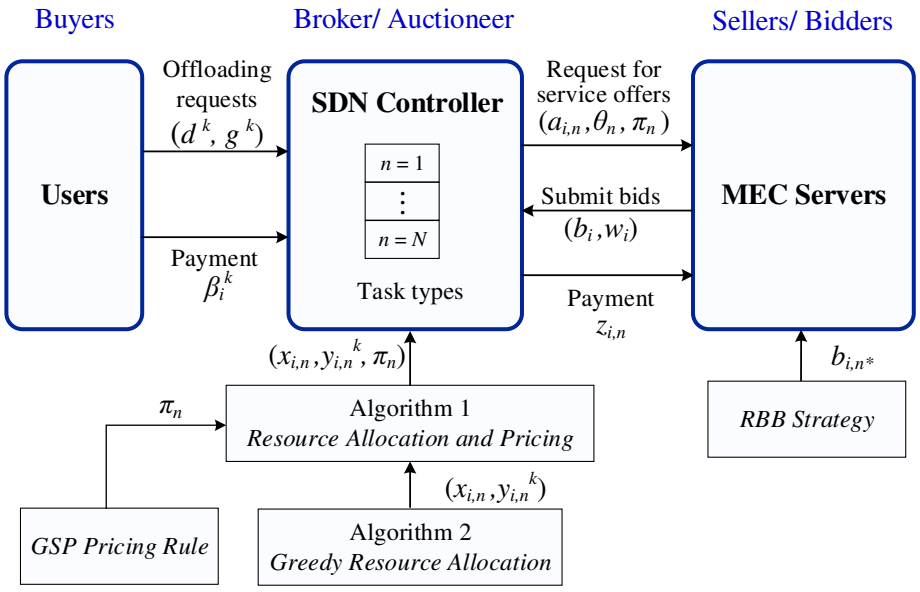


Figure 3.2: Proposed resource allocation and pricing mechanism

users demands. With the goal of determining the winning bidders, the SDN controller first sorts the bidders in the decreasing order based on their bid efficiency, i.e. the ratio of total value to the total weight that an MEC server can offer for computing an offloading task. I define the effective bid for MEC server i as

$$\eta_i = \log\left(\hat{a}_i/\hat{b}_i\right), \tag{3.10}$$

where $\hat{a}_i = \sum_{n=1}^{N}\sum_{k=1}^{K} a_{i,n}^k$ and $\hat{b}_i = \sum_{n=1}^{N}\sum_{k=1}^{K} \gamma_i^k\, \theta_n b_{i,n} w_i$. Note that in the denominator of (3.10), the valuation of the bidder for each task type is scaled according to the users' satisfaction level γ_i^k. This approach ensures that bidders with higher satisfaction indices are given preference over those who failed to satisfy the users so far. Moreover, in (3.10), I use the logarithmic function that transforms the larger values of effective bids in logscale.

Next, the SDN controller formulates the following optimization problem, referred to as the winner determination problem (WDP) to deter-

mine the winning bidders who get offloading tasks for each task type $n \in \mathcal{N}$:

$$
\begin{aligned}
\mathcal{P}1: \quad & \max_{x_{i,n}, y^k_{i,n}} V = \sum_{i=1}^{I} \eta_i \, x_{i,n} \\
\text{s.t.} \quad (\text{C1}) \quad & \sum_{i=1}^{I} x_{i,n} \geq \sum_{i=1}^{I} \sum_{k=1}^{K} a^k_{i,n} \, y^k_{i,n}, && \forall \, n \in \mathcal{N}, \\
(\text{C2}) \quad & \sum_{n=1}^{N} x_{i,n} \leq w_i, && \forall \, i \in \mathcal{I}, \\
(\text{C3}) \quad & \sum_{i=1}^{I} y^k_{i,n} \leq 1, && \forall \, n \in \mathcal{N}, k \in \mathcal{K}, \\
(\text{C4}) \quad & x_{i,n} \in \{0, 1, \ldots, w_i\}, && \forall \, i \in \mathcal{I}, n \in \mathcal{N}, \\
(\text{C5}) \quad & y^k_{i,n} \in \{0, 1\}, && \forall \, i \in \mathcal{I}, n \in \mathcal{N}, k \in \mathcal{K}, \quad (3.11)
\end{aligned}
$$

.

Here, the objective is to select MEC servers who provide computation service at lower price and to allocate their resources in a way that maximize users' level of satisfaction, i.e. total valuation of effective bids. The constraint (C1) ensures that total amount of allocated resources meets users' the resource requirements. The constraint (C2) ensures that total amount of allocated resources does not exceed the assigned MEC server's available resource capacity. The constraint (C3) ensures that an offloading task is assigned to at most one MEC server. The constraints (C4) and (C5) state weight/values of the decision variables $x_{i,n}$ and $y^k_{i,n}$, respectively. The mixed integer program $\mathcal{P}1$ is a variant of generalized assignment problem with minimum quantities which is NP-complete [23]. However, it could be solved optimally in

polynomial time if the profit of assigning a task is independent of the edge server it is assigned to and the maximum resource capacity of the edge servers as well as number of offloading tasks are fixed. The computational complexity of $\mathcal{P}1$ also comes from the fact that for a given offloading data size, the amount of CPU resources required from each MEC server is different. To simplify the computation, the integer constraints can be relaxed as $0 \leq x_{i,n} \leq w_i$ and $y^k_{i,n} \geq 0$ to allow fractional amount of resource allocation and derive the upper bound of $\mathcal{P}1$. Then, the optimal solution can be obtained by solving the upper bound of $\mathcal{P}1$ using the classic *Branch and Bound* technique. Although such procedure involves high computational complexity, commercial solvers (e.g. CPLEX) could be used for practical implementation. In the following section, I propose an approximation algorithm with polynomial solution time.

3.3.3 Resource Allocation and Pricing Algorithm

I propose an approximation algorithm based on the greedy heuristic approach in combinatorial optimization. The proposed resource allocation and pricing mechanism is depicted in Fig. 3.2. The SDN controller performs the mechanism during each offloading period. **Algorithm 1** summarizes the solution steps of the proposed resource allocation and pricing approach.

I define K_n as the set of users who request for task type n. Moreover, $\bar{w}_i$ is the remaining capacity of MEC server i in terms of the number of virtual CPU resource units. As described in the algorithm, the SDN controller first calculates the effective bids, given the MEC servers submitted bids and users' feedback. Then, it rearranges the MEC server

Algorithm 1: *Resource Allocation And Pricing*

Input: d^k, g^k, b_i, w_i
Output: $V, x_{i,n}, y^k_{i,n}, \pi_n$

$V \leftarrow 0, \mathbf{x}_{I\times N} \leftarrow 0, \mathbf{y}_{I\times N\times K} \leftarrow 0, \pi_N \leftarrow 0$
for $i \leftarrow 1$ **to** I **do**
    Calculate γ^k_i using (3.6)
    Calculate η_i using (3.10)
end
Rearrange the MEC servers as $\eta_1 \geq \eta_2 \geq \cdots \geq \eta_N$
for $n \leftarrow 1$ **to** N **do**
    Let $K_n = K_n \cup \{k \,|\, g^k = n\}$
    Rearrange the users in K_n as $\lambda^n_1 \geq \lambda^n_2 \geq \cdots \geq \lambda^n_K$
    Let $\bar{w}_i = w_i$
    Call Algorithm 2: *Greedy Resource Allocation*
end
Find allocation prices, π_n, using eqn. (3.12)

indices in a decreasing order of η_i. Afterward, the SDN controller considers the offloading tasks one by one according to the task types. For each task type n, it calls the greedy procedure in **Algorithm 2**. The outcome is the assignment of tasks to the available MEC servers one by one unless the remaining capacity is smaller than the resource demand $a^k_{i,n}$. For each feasible allocation, the algorithm updates the following parameters: (i) $y^k_{i,n} = 1, x_{i,n} = x_{i,n} + a^k_{i,n}$, (ii) $V = V + \eta_i\, x_{i,n}$, and (iii) $\bar{w}_i = \bar{w}_i - a^k_{i,n}$.

When **Algorithm 2** returns the task assignment decision $y^k_{i,n}$, resource allocation decision $x_{i,n}$, and the auction welfare V to **Algorithm 1**, the allocation prices are determined according to the GSP rule of position auction. I consider π_n as the unit resource price for the assignment of an MEC server to tasks of type n. By the GSP rule, π_n is determined according to the bid of the next bidder who wins $n+1$. I define i_0 is the index of the last MEC server who wins any task of type n. Moreover, i' is the MEC server ranked next to the bidder i_0

according to decreasing order of effective bids.

The allocation price for task type n is determined as follows:

$$\pi_n = \begin{cases} b_{i',n}, & \text{if } 1 \leq n \leq N, rank(i') \leq I, \\ b_{i_0,n} + \epsilon, & \text{otherwise} \end{cases} \tag{3.12}$$

where ϵ is a very small positive real number. The bidders who are not assigned to any task type receive no payment and thus they have no utility gain.

Algorithm 2: *Greedy Resource Allocation*

Input: $N, K_n, \bar{w}_i, a_{i,n}^k$
Output: $V, x_{i,n}, y_{i,n}^k$

$V \leftarrow 0, \mathbf{x}_{I\times N} \leftarrow 0, \mathbf{y}_{I\times N\times K} \leftarrow 0$
for $k \in K_n$ **do**
  if $(l_k = 0)$ **AND** $(a_{i,n}^k + x_{i,n}) \leq \bar{w}_i)$ **then**
    Let $y_{i,n}^k = 1, x_{i,n} = x_{i,n} + a_{i,n}^k, V = V + \eta_i\, x_{i,n}$
    Let $\bar{w}_i = \bar{w}_i - x_{i,n}$
  end
end

3.4 Analysis on Auction Efficiency

The proposed GSP-based reverse auction framework is economically robust by satisfying the individual rationality and envy-free properties. The following propositions show the stability and efficiency of the proposed resource allocation mechanism.

Proposition 1 *The proposed reverse auction mechanism is individually rational, this means that if MEC servers bid according to RBB strategy, they would be guaranteed with non-negative utilities for each winning task type, i.e.* $u_{i,n} \geq 0$.

Parameters and values	
$N = 3, \quad I = 5 \quad B_i = [5, 30]$ MHz	$d^k_{\max} = [100, 500]$ KB
$L_1 = 2640, L_2 = 1760, L_3 = 8250$	$p^k = 20$ dBm, $p_i = 40$ dBm
$\theta_1 = 0.6, \theta_2 = 0.3, \theta_3 = 0.1$	$\delta^k_{\max} = \{300, 500, 1000\}$ msec
$M_i = \{2, 4, 8, 16\}$ with $f_i[2, 4]$ GHz/core	$\beta^k_{\max} = 10,\ \alpha^a_k = 1$
$\sigma^2 = -100$ dBm, $P_0 = 0.01$ watts	$\alpha^d_k = 0.005,\ \alpha^p_k = 0.1$
$\rho_i = [0.002, 0.004]$ \$ / CPU resource unit	$\alpha^e_i = 0.001,\ \kappa_i = 0.01$

Table 3.2: System parameters used for simulation in Chapter 3

Proof. See Appendix A.1. ■

Proposition 2 *The proposed reverse auction mechanism is locally envy-free with allocation prices $\pi_n, \forall n \in \mathcal{N}$. This characteristic guarantees that no bidder can improve her utility by exchanging her allocation with any other bidder for the same task type.*

Proof. See Appendix A.2. ■

Proposition 3 *The proposed GSP-based reverse auction mechanism is computationally efficient, i.e., approximation solution to WDP in $\mathcal{P}_1$ can be obtained in polynomial time.*

Proof. See Appendix A.3. ■

3.5 Numerical Results

I consider an MEC offloading scenario where the number of users K varies from 100 to 1000 over different offloading periods. The MEC servers and users are randomly deployed in 5 km^2 area. I use the distance-dependent path-loss model $-140.7 - 36.7 \log_{10}$(distance to find the channel power gain h^k between users and their BSs. I run the simulation for 1000 times. During each run, each user randomly requests

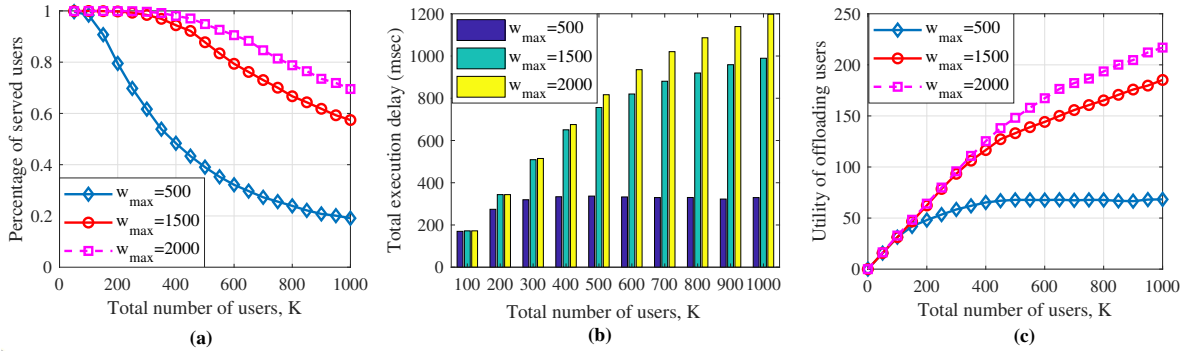


Figure 3.3: Performance of the proposed algorithm: Percentage of served users, total execution delay of all the served users, and their utilities in terms of satisfaction functions.

one of the $N = 3$ task types. The offloading data size is random and belongs in the interval $[0, d^k_{\max}]$. The simulation parameters used for analysis are listed in Table 3.2.

I first investigate the users' satisfaction level. The results in Fig. 3.3 (a) show that the percentage of the served user decreases as the number of users rise, due to limited resource capacities of the servers. If the MEC servers can offer more resources, e.g. $w_{\max} = 2000$, the number of the served users would increase. Fig. 3.3(b) shows the increase in total execution time for larger number of served users. The overall satisfaction level of the served users is shown in Fig. 3.3(c). It can be concluded that when the offered computing capacities is higher, specifically for $w_{\max} = 1500$ and 2000, the sum utilities of the served users is significantly higher than that of with $w_{\max} = 500$. This results from the users experiencing longer delays when MEC servers have low computing capacity.

Next, I evaluate the performance of the proposed algorithm in solving the WDP. Fig. 3.4 (a) shows the objective function of WDP V, which increases with the increase in number of offloading users. Such trend

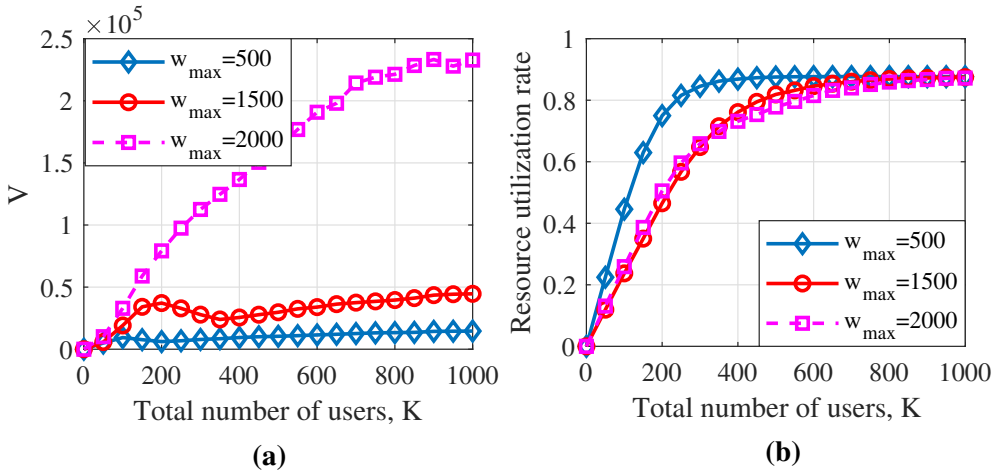


Figure 3.4: Performance of proposed algorithm on WDP and the system's resource utilization rate.

implies that the bidders' offered resources are allocated efficiently as to satisfy user demands. It should be also noted that changing $w_{\max}$ (the maximum availability of computing resource units for MEC servers), impacts the total valuation of the WDP solution dramatically. Moreover, the valuation of the WDP solution is low for $w_{\max} = 500$ compared to other tested values. The reason is the following: In this setting, the MEC servers compete over a low number of offloading tasks and thus the efficiency of bids decreases. I define the *rate of resource utilization* as the ratio of the total amount of allocated resource units to the total amount of available resource units, considering the maximum user capacity of the system is $K_{\max} = 1000$. Then, from Fig. 3.4(b), although the system's efficiency is low for smaller number of users, it gradually reaches the 90% of resource utilization.

3.6 Conclusion

In this chapter, have designed a reverse auction model for an MEC offloading system, and formulated the winner determination problem (WDP) as a combinatorial optimization problem. To solve the WDP, I have designed an approximation algorithm that determines resource allocation and prices in polynomial time. I have also proposed an adaptive greedy bidding strategy that allows the MEC servers to o maximize their utilities. Finally, I have demonstrated the auction efficiency and performance of our proposed auction mechanism through theoretical and numerical analysis.

Chapter 4

A Repeated Auction Model for Load-Aware Dynamic Resource Allocation in MEC

In this chapter, I investigate MEC offloading service provisioning addressing the wireless network dynamics and competition among multiple MEC service providers. Considering dynamic arrival of offloading requests and computation workload-dependent computational performance metrics for the servers, I present a service-oriented MEC offloading architecture, and apply repeated GSP auction model to develop an efficient resource allocation and pricing mechanism.

In Section 4.1, I briefly discuss the research problem I address in this chapter and summarize the contributions. In Section 4.2, I outline the MEC offloading service provisioning system model and assumptions, and also describe the overall MEC service provisioning and communication workflow. Section 4.3 presents the repeated GSP-based load-aware dynamic resource allocation model and Section 4.4 provides the analysis of competitive bidding behavior of the MEC servers. Finally, I present the numerical results in Section 4.5, and conclude the chapter in Section 4.6.

4.1 Introduction

In a multi-user multi-vendor MEC offloading market environment, efficient resource allocation and pricing strategies are vital, since the users' hetereogeneous QoS demands needs to be met using the limited resources of the servers at a minimal offloading cost. At the same time, is it crucial to guarantee that no service providers encounter any profit loss, so they are motivated to deploy various wireless applications via MEC service delivery platform. The deployment of MEC services involves several challenges within the wireless network environment itself. Thus it becomes more challenging to ensure allocation efficiency along with economic benefits, addressing the competition among the resource sellers.

Firstly, task arrivals are highly dynamic in the wireless environment as computing resource demands and QoS requirements evolve. Hence, it is challenging to accurately model the uncertainties, including task arrivals and QoS criteria. Secondly, the computing resources at the MEC servers are scarce, and the computational workloads of these servers are highly variable due to the heterogeneity in offloading tasks' length and computational capacity requirements of different services. That heavily affects the offloading performance. Besides, self-interested MEC service providers design their bidding strategies to sell more resource and maximize their revenue.

The auction game theory offers promising tools to address efficient resource allocation and pricing policies, and model the offloading service provisioning process considering the demand-supply trend and fairness. In order to address the challenges in provisioning MEC services

in the dynamic environment, I present a novel service-oriented MEC offloading system architecture that deploys the offloading as a service through a repeated GSP auction mechanism. The proposed repeated auction model supports the network dynamics and allocates resources to offloading users accordingly. In order to guarantee the stability of allocation outcomes in such a dynamic offloading scenario, I investigate the adaptive best-response bidding strategies and corresponding equilibrium pricing strategies, so that economic properties are satisfied for the proposed repeated auction mechanism.

4.1.1 Research Contribution

The key contributions of this chapter are as follows:

- I develop a generic service-oriented multi-user multi-vendor MEC system architecture suitable for SDN-enabled 5G networks.
- I present computational workload-aware resource management policies, which implement dynamic resource allocation through a repeated GSP auction mechanism.
- In addition, I propose best-response restricted balanced bidding (RBB) strategies for the resource sellers (i.e., MEC servers), that ensures equilibrium resource allocations under a dynamic environment.
- I provide theoretical analysis on competitive bidding behavior of MEC servers, and presents numerical results to evaluate the performance of the proposed auction mechanism.

4.2 System Model

4.2.1 Wireless Network and Communication Model

Considering I MEC sites gathered in the set $\mathcal{I} = \{i\}_{i=1}^{I}$, I assume each site is equipped with a WAP, a computing server, along with a site controller node. I assume that different computation service providers operate each site while competing to earn more revenue by providing computation offloading as services for N applications. The physical servers at the MEC sites have SDN functionalities to host N application processes simultaneously. I consider a centralized SDN hypervisor, referred to as the *MEC orchestrator* or *orchestrator*. The orchestrator coordinates all the control functionalities across the MEC sites and deploys the computation offloading service between MEC servers and UEs.

Let $\{j\}_{j=1}^{J}$ be the set of UEs uniformly distributed across the MEC sites, and each UE j is associated with the nearest WAP $i' \in \mathcal{I}$. The UEs get exclusive OFDMA sub-carriers to transmit on the wireless links without interference. For simplicity, I assume the UEs are stationary so the user association remains fixed. Therefore, the same WAP handles all the offloading requests from a user on the MEC site. However, when the offloading requests are forwarded to the MEC system, they can be processed at a different MEC site depending on the tasks' QoS requirements and the server's computational capabilities.

The offloading data rate (in MBps) in the uplink between the UE j and the associated WAP i' is given by

$$\gamma_{i',j} = \mathrm{BW}_{i'} \log_2 \left(1 + \frac{P_j^{\mathrm{up}} h_{i',j}}{\sigma_N^2}\right), \tag{4.1}$$

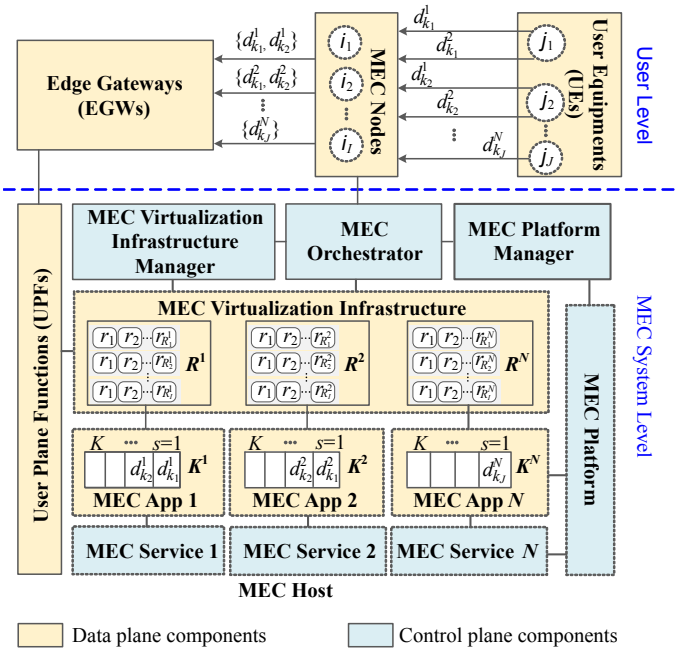


Figure 4.1: Service-based MEC system architecture.

where σ_N^2 is the noise variance, $\mathrm{BW}_{i'}$ the bandwidth of the channel assigned by WAP i', and P_j^{up} the transmit power of UE j.

I consider a generic transmission loss model [37], assuming both UEs and WAPs are below the rooftop level regardless of their antenna heights. Thus the basic transmission loss (in dB) for short-range outdoor communication yields

$$h_{i',j} = 10\mu_d \log_{10}\left(\mathrm{dist}_{i',j}\right) + \mu_0 + 10\mu_f \log_{10}\left(f_t\right) \tag{4.2}$$

where $\mathrm{dist}_{i',j}$ (in meter) is the distance between UE j and WAP i', and f_t is the wireless channel's operating frequency. Besides, μ_d and μ_f are the coefficients that describe the growth of transmission loss with distance and frequency, respectively. Also, μ_0 is the coefficient associated with the offset value of the basic transmission loss.

Table 4.1: List of key notations used in Chapter 4

Notation	**Description**
N	Number of MEC application processors
J	Number of offloading UEs
I	Number of MEC servers
$\gamma_{i',j}$	Uplink data rate between UE j and MEC node i'
$\mathcal{R}^n$	Set of VMs at processor n
R_i^n	Number of VMs of type n at server i
r_{im}^n	m-th VM of type n at server i
$\mathcal{C}_i^n$	Computing power of each VM of type n at server i
W_i^n	Number of vCPUs in each VM of type n at server i
β^n	Computational capacity requirements of each task in processor n
η_{i,r_m}^n	Workload (MB) of the m-th VM of type n at server i
Γ_{i,r_m}^n	Load per capacity of the m-th VM of type n at server i
ϕ_{i,r_m}^n	Resource utilization rate of the m-th VM of type n at server i
θ_{i,r_m}^n	Expected quality score of the m-th VM of type n at server i
v_{i,r_m}^n	Valuation of the m-th VM of type n at server i
b_{i,r_m}	Bid submitted by server i for the m-th VM of type n
y_{i,r_m}^n	Ranking score of the m-th VM of type n at server i
$\mathcal{K}^n$	Task queue with $K = J$ positions at processor n
k_j^n	Computing task of type n offloaded by UE j
$d_{k_j}^n$	Length of task (MB) offloaded by UE j to processor n
$\lambda_{s_j}^n$	Task priority index of UE j in processor n
$x_{s,r_{im}}^n$	Offloading task-VM matching decision variable
p_s^n	Allocation price (\$/VM-hour) decision variable
u_{i,r_m}^n	Utility gain of the m-th VM of type n at server i

4.2.2 MEC Service Provisioning Model

I model a service-oriented MEC system architecture [35] to implement computation offloading as a service. As depicted in **Figure 4.1**, the orchestrator is in charge of the overall computation offloading service provisioning process. It starts by receiving offloading requests from the UEs to allocate computing resources to process the offloaded tasks and manage corresponding resource allocation payments. The orchestrator interacts with the MEC sites, EGWs, and UEs through control and user plane functions (UPFs), and implements the offloading services with the help of two other SDN controllers: (a) MEC virtualization infrastructure manager and (b) MEC platform manager.

The *virtualization manager* mainly oversees the computing infrastructure resources (i.e., servers) of the MEC system. It abstracts and partitions the computational resources into virtualized CPU (vCPU) resource units using SDN functionalities. For each server i, the virtualization manager defines N different sets of virtual machine (VM) instances and then allocates vCPUs into these instances aligning with the computational processing requirements for N MEC applications/ services. As shown in **Figure 4.1**, each processor n has a resource pool consisting of $R^n = \sum_{i=1}^{I} r_i^n$ VMs, where R_i^n indicates the number of VMs available at server i to process the tasks of application n. I represent the m-th VM at server i by r_{im}^n, which has computing power C_i^n (MBps) and W_i^n vCPUs each with CPU frequency $f_{C_i}^n$ (in Hz).

The *platform manager* supervises the offloading task execution process by managing the N MEC application processors via the MEC platform. The platform maintains a task queue $\mathcal{K}^n$ for each processor

n to handle the incoming offloading requests for application n. Each queue $\mathcal{K}^n$ has a fixed ($K = J$) number of positions. Each position s is associated with a user-specific task priority index $\lambda^n_{s_j}$ that helps prioritize the requests.

To complete each offloading task of application type n is within the deadline $\tau^n_{\max}$ (msec), the minimum average processing speed (MBps) for the UE j's request yields $f^n_{C_{\min}} \geq (\bar{d}^n_j/\tau^n_{\max})$ [2]. Therefore, I consider a delay-aware task prioritizing policy defining the task priority index for the s-th task position as

$$\lambda^n_{s_j} = \frac{\bar{d}^n_j}{\tau^n_{\max}} \tag{4.3}$$

that prioritizes offloading requests with larger data sizes and shorter task completion deadlines. The platform manager handles the incoming offloading requests according to the requested task type n. The platform manager forwards the requests to the corresponding processor n and places them into the task queue $\mathcal{K}^n$ according to the requesting UE j's priority index $\lambda^n_{s_j}$. The orchestrator then coordinates the auction process that matches the tasks and suitable VMs. I consider non-preemptive priority-based task assignments at each processor; that means allocated VMs are not released until task processing finishes.

4.2.3 MEC Orchestration Model

The orchestrator manages the end-to-end computation offloading service provisioning process through an auction. I model the orchestration process as a dynamic position auction game between two sets of players: UEs and MEC servers, while the orchestrator is the auctioneer. It runs the auction mechanism at discrete offloading time slots, $t = 1, 2, \ldots,$

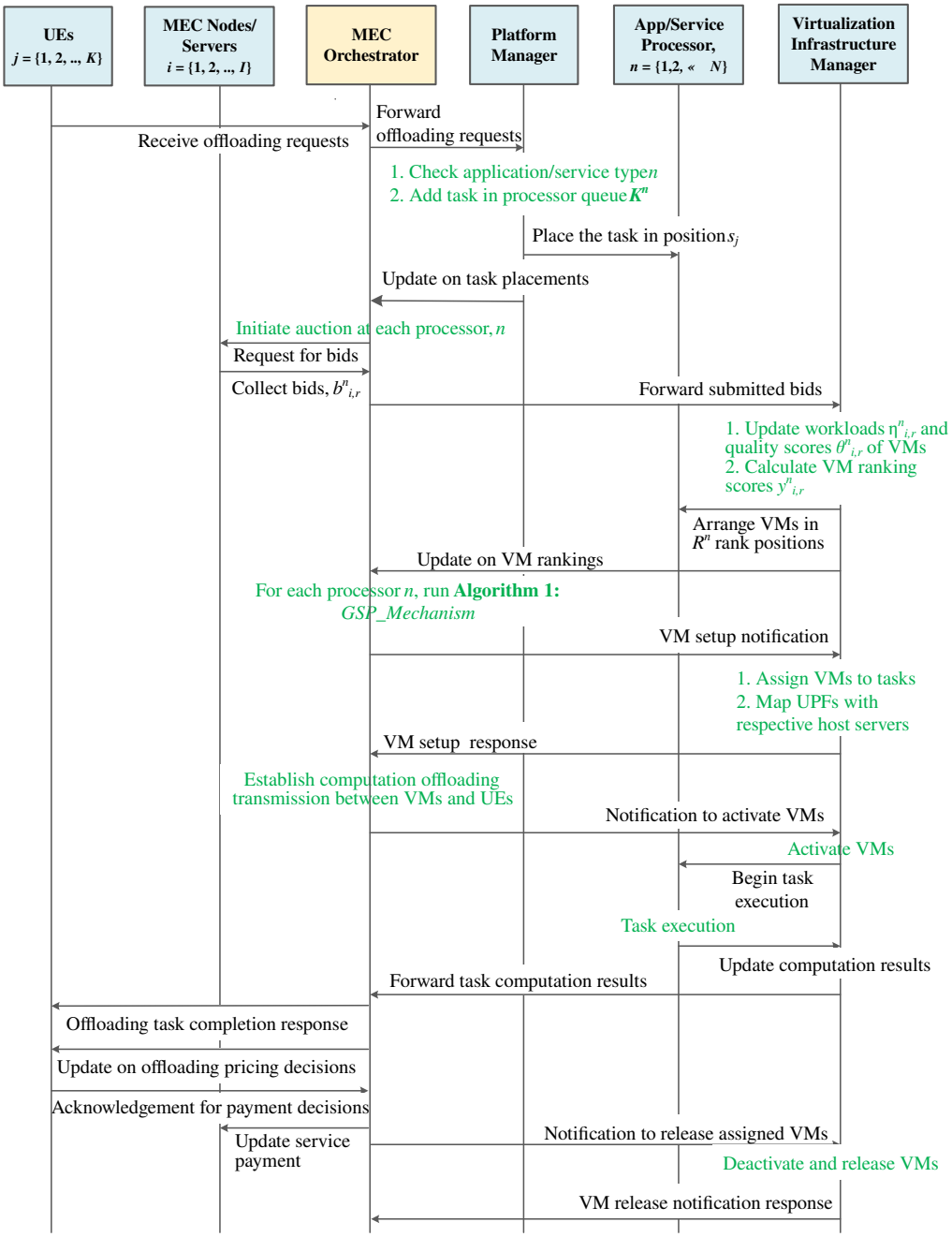


Figure 4.2: Workflow of the computation offloading mechanism in MEC framework.

with a duration of Δt, at N processors. Thus, it obtains N sets of independent allocations. For each auction round in processor n, $K = J$ UEs are the resource buyers. The I servers are then the sellers with R^n VMs as the auction commodities.

Formally, at processor n and time slot $t > 0$, I express the offloading service provisioning state as $\langle \mathcal{K}^n(t), \mathcal{R}^n(t), \mathcal{X}^n(t), \mathcal{P}^n(t) \rangle$, where

- $\mathcal{K}^n(t)$ represents the task queue with K distinct positions. When the UE j's request arrives at the processor n, its computing task k_j^n with length $d_{k_j}^n$ (MB) takes position $s_j \in \mathcal{K}^n(t)$ with the priority index $\lambda_{s_j}^n$,

- $\mathcal{R}^n(t)$ represents the list of ranked VMs in processor n, where each VM r_{im}^n has a position m. The position is updated every time slot based on the current computing processing score $\theta_{i,r_m}^n(t)$ and bid $b_{i,r_m}^n(t)$,

- $\mathcal{X}^n(t)$ represents the offloading task-VM matching decisions, where $x_{s,r_{im}}^n(t) \in \{0, 1\}$ is the decision variable indicating if the task in position s is assigned to the m-th VM at server i in time slot t,

- $\mathcal{P}^n(t)$ represents the allocation pricing decisions. The decision variable $p_s^n(t) \geq 0$ indicates the amount (in \$/VM-hour) that the UE pays for offloading the task in position $s \in \mathcal{K}^n(t)$.

I describe the orchestration stages below and summarize the overall workflow in **Figure 4.2**.

(i) **Dynamic Queuing of Incoming Offloading Tasks**: At the beginning of time slot t, the orchestrator gathers the incoming

offloading requests and forwards them to the platform manager. The platform manager checks their task types and arranges them into respective processing queues $\mathcal{K}^n(t)$. The offloading requests arrive dynamically, i.e., the orchestrator does not have any prior information about the number of tasks in the current time slot until the requests arrive.

(ii) **Dynamic Resource Management Based on Computational Workloads**: The virtualization manager uses dynamic resource management policies and maintains separate queues to monitor the computational workloads of VMs at each processor n. At the beginning of every time slot t, it checks the status of VMs and updates their workloads in the queue denoted by $\eta^n_{i,r_m}(t)$. The workload of each VM queue thus evolves as

$$\eta^n_{i,r_m}(t) = \eta^n_{\text{rem},i,r_m}(t-1) + \sum_{s=1}^{K} d^n_{k_j}(t) x^n_{s,r_{im}}(t) \tag{4.4}$$

where

$$\eta^n_{\text{rem},i,r_m}(t-1) = \max\left\{ \sum_{s=1}^{K} d^n_{k_j}(t) x^n_{s,r_{im}}(t-1) - C^n_i \Delta t, 0 \right\}$$

represents the remaining workload of the VM r^n_{im} after computing the task assigned in round $(t-1)$. Therefore, the current computational workload per capacity (Byte per CPU cycle (BPC)) on the m-th VM at server i can be estimated as [48]

$$\Gamma^n_{i,r_m}(t) = \frac{\eta^n_{i,r_m}(t)}{C^n_i \Delta t}. \tag{4.5}$$

Based on the VM's workload per capacity, the resource utilization

metric follows as

$$\phi^n_{i,r_m}(t) = \begin{cases} 0, & \text{if } \Gamma^{\max} \leq \Gamma^n_{i,r_m}(t) \\ \dfrac{\left|\Gamma^n_{i,r_m}(t) - \Gamma^{\max}\right|}{\Gamma^{\max}}, & \text{if } \Gamma^{\min} \leq \Gamma^n_{i,r_m}(t) < \Gamma^{\max} \\ 1, & \text{otherwise} \end{cases} \tag{4.6}$$

where $\Gamma^{\max}$ indicates the maximum load allowed per capacity on each VM. Task assignment beyond this limit would overload the VM with the lowest utilization score (i.e., $\phi^n_{i,r_m} = 0$). In contrast, it scores the highest (i.e., $\phi^n_{i,r_m} = 1$) when the VM is underloaded, i.e., the existing load is smaller than the minimum resource utilization threshold $\Gamma^{\min}$; otherwise, the scoring function determines the resource utilization scores for the VMs under normal workload between 0 and 1.

The expected computation performance quality score of a VM is updated in each time slot t, according to its current resource utilization as

$$\theta^n_{i,r_m}(t) = \frac{W^n_i f^n_{C_i}}{f^n_{C_{\min}}} \phi^n_{i,r_m}(t). \tag{4.7}$$

(iii) **Collecting Bids from Servers**: After queuing the incoming offloading requests and updating VMs' expected computation performance quality score, the orchestrator initiates the auction with the bid collection process by requesting servers to submit bids for each processor n. Also, it provides the servers with information on service provisioning state values from the previous round of the auction.

The servers determine their bids for each round, following their best-response bidding strategies (as I discuss in **Section 4.4**). For

each server i, I use $\mathbf{b}_i^n(t) = \left[b_{i,r_1}^n, b_{i,r_2}^n, \ldots, b_{i,r_{R_i^n}}^n\right]$ to denote its bids for its VMs in processor n in round t.

(iv) **Determining Resource Allocation Decisions**: After receiving the updated bids, the orchestrator runs the resource allocation and pricing algorithm at each processor n. I consider GSP-based resource allocation and pricing rules to determine the task assignment decisions, $x_{s,r_{im}}^n(t)$, and VM allocation pricing decisions, $p_s^n(t)$. I outline the resource allocation and pricing in **Algorithm 3**.

(v) **Executing Offloading Tasks at Servers**: In the next stage, the orchestrator notifies the virtualization manager to set up the VMs according to the task-VM matching decisions. The virtualization manager then maps the VMs and tasks to the host servers and UPFs, respectively. It updates the orchestrator after setting up the VMs and then activates the VMs to execute tasks after receiving the notification from the orchestrator. In the end, it sends the computed results back to the UEs via the orchestrator.

After sending the task computation results, the orchestrator updates the UEs about the total offloading service payment information and notifies the virtualization manager to release the assigned VMs. Once the UEs acknowledge the service billing, the orchestrator ends the current offloading transmission session. I consider *pay-per-CPU cycle* payment methods for processing offloading tasks, where payments are collected at the end of the billing cycle (e.g., bi-weekly, monthly, annually).

4.2.4 Utility Model of MEC Servers

I consider the utility model for the servers based on the profits earned from VM allocation in each auction round. The profit for each VM is given by the price difference between the allocation price settled by the auctioneer (orchestrator) and the VM's private valuation.

Let v^n_{i,r_m} be the private valuation (\$/VM-hour) of the m-th VM at server i in processor n, and is determined based on the CPU power consumption at the processor [62],

$$v^n_{i,r_m} = \rho_i \kappa W^n_i \left(f^n_{C_i}\right)^2 \tag{4.8}$$

where ρ_i is a scaling parameter to convert the CPU power consumption into monetary value. Besides, κ is the effective switched capacitance of the processor.

Therefore, if server i allocates m VMs to task position s in processor n in time slot t, its profit follows as

$$u^n_{i,r_m}(t) = \sum_{s=1}^{K} \lambda^n_{s_j} \theta^n_{i,r_s}(t) \left(p^n_s(t) - v^n_{i,r_s}\right) x^n_{s,r_{i(m=s)}}(t) \tag{4.9}$$

The total utility of server i in time slot t thus yields

$$U_i(t) = \sum_{n=1}^{N} \sum_{m=1}^{R^n_i} u^n_{i,r_m}(t). \tag{4.10}$$

4.2.5 Utility Model of Offloading Users

The utility of the offloading users depends on their QoE in terms of both task offloading cost and task execution latency. The total execution time of task k^n_j offloaded by UE j consists of (i) upload, (ii) queue at the MEC platform, (iii) computation at the allocated VM, and (iv)

sending back the results. Often, the size of the computed results is negligible compared to uploading the data; Hence, I ignore the time to send back the results in the downlink. Therefore, the end-to-end offloading service latency (in sec) for the task k_j^n in time slot t can be written as

$$\delta_{k_j}^n(t) = \delta_{k_j,\text{up}}^n(t) + \delta_{k_j,\text{wait}}^n(t) + \delta_{k_j,\text{comp}}^n(t)\,, \tag{4.11}$$

where for UE j, the upload time to MEC node i' is given by $\delta_{k_j,\text{up}}^n(t) = \frac{d_{k_j}^n(t)}{\gamma_{i',j}}(t)$. At server i, the computation time is $\delta_{k_j,\text{comp}}^n(t) = \frac{d_{k_j}^n(t)}{C_i^n}$. The waiting latency for each request depends only on the computation time of the tasks placed ahead in the task queue. Thus, for a task in position $s \in \mathcal{K}^n(t)$, it can be estimated as $\delta_{k_j,\text{wait}}^n(t) = \sum_{s'=1}^{s-1} \delta_{k_{s'_j},\text{comp}}^n(t)$, where $k_{s'_j}$ denotes the task in an upper position ($s' < s$) in the queue.

To quantify UE j's level of satisfaction with the overall service latency for a task k_j^n, I define the following performance metric:

$$\alpha_{k_j}^n(t) = \begin{cases} \dfrac{|\tau_{\max}^n - \delta_k^n(t)|}{\tau_{\max}^n}, & \text{if } 0 < \delta_{k_j}^n \leq \tau_{\max}^n \\ 0, & \text{otherwise} \end{cases} \tag{4.12}$$

Therefore, UE j's QoE in terms of offloading service latency can be estimated as the following mean opinion score

$$Q_j^{\text{latency}}(t) = \frac{1}{N}\left[\sum_{n=1}^{N}\sum_{s=1}^{K}\sum_{r=1}^{R^n} \alpha_{k_j}^n(t)\, x_{s,r_{im}}^n(t),\right] \tag{4.13}$$

Next, considering $\bar{a}_j$ (\$) as the UE j's monetary budget, I model the UE's QoE in terms of offloading service cost using the budget cost savings ratio as

$$Q_j^{\text{cost}} = \left|\frac{\bar{a}_j - a_j(t)}{\bar{a}_j}\right|, \tag{4.14}$$

where $a_j(t)$ represents the cost (in \$) to process the offloading tasks, which depends on the processing time spent by the allocated VMs. Formally,

$$a_j(t) = \sum_{n=1}^{N} \sum_{s=1}^{K} \sum_{r=1}^{R^n} \delta^n_{k_j,\text{comp}}(t)\, p^n_s(t)\, x^n_{s,r_{im}}(t) \tag{4.15}$$

The overall utility gain of the UE j thus becomes

$$Q_j = q_l\, Q_j^{\text{latency}} + q_c\, Q_j^{\text{cost}}, \tag{4.16}$$

where $q_l \in [0,1]$ and $q_c \in [0,1]$ are the QoE coefficients to adjust the trade-off between the offloading service latency and offloading cost, respectively.

4.3 GSP-Based Auction Mechanism Design for Dynamic Computation Offloading and Resource Allocation

I develop a GSP-based auction mechanism for offloading service provisioning at N MEC processors at every time slot t. Indeed, the orchestrator performs the task assignment and resource allocation using the auction mechanism. The goal is to maximize the total valuation of the allocated VM resources in each time slot.

I first formulate the winner determination problem (WDP) by considering an offline version of computation offloading as a snapshot. Then, I study the WDP solutions and decide on resource pricing at the current auction round. Later, I describe the repeated auction mechanism enabling the orchestrator to allocate the resources for each processor n in any time slot t in a dynamic computation offloading setting.

4.3.1 Winner Determination Problem Formulation

I consider a static computation offloading scenario at the n-the processor. There are K computing tasks in the queue $\mathcal{K}^n$ and I servers, each having a distinct set of VM resources, $\{R_i^n\}$. Given the quality scores of the VMs, θ_{i,r_m}^n, and the bids submitted by the servers, b_{i,r_m}^n, the orchestrator formulates the following optimization problem that maximizes the total allocation valuation by determining the winners (i.e., VMs) for each processor n.

$$
\begin{aligned}
\max \quad & \sum_{s=1}^{K} \sum_{i=1}^{I} \sum_{r_{im} \in R_i^n} z_{s,r_{im}}^n \, x_{s,r_{im}}^n \\
\text{s.t.} \quad & \text{(C1)} \ \sum_{i=1}^{I} \sum_{r_{im} \in R_i^n} x_{s,r_{im}}^n = 1, \quad \forall \, s \\
& \text{(C2)} \ \sum_{s=1}^{K} \sum_{r_{im} \in R_i^n} x_{s,r_{im}}^n \leq R_i^n, \quad \forall \, i \\
& \text{(C3)} \ \ x_{s,r_{im}}^n \in \{0,1\}, \, \forall \, s, r_{im}
\end{aligned}
\tag{4.17}
$$

where $z_{s,r_{im}}^n = \lambda_{s_j}^n \theta_{i,r_m}^n / b_{i,r_m}^n$ represents the allocation valuation for the VM $r_{im} \in R_i^n$, who wins the s-th offloading task. In (4.17), constraint (C1) guarantees that each offloading task is matched with exactly one VM. The constraint (C2) ensures that the total number of tasks assigned to a server i does not exceed its VM resource constraints. Furthermore, (C3) means that the offloading task assignments are binary decision variables.

4.3.2 WDP Solution Approaches

The orchestrator determines the corresponding resource allocation prices, $\{p_s^n\}$, based on the offloading task assignment decisions, $\{x_{s,r_{im}}^n\}$ ob-

tained through solving the WDP in (4.17). To achieve socially-efficient allocation outcomes, one can adopt the classic VCG pricing mechanism [55] so that the allocation algorithm solves the WDP by selecting the VMs that gives maximum allocation valuation for each offloading task. It then settles the allocation prices for each bidder equivalent to the amount it contributes to social welfare. However, the VCG mechanism is often unsuitable for practical auction design, especially for time-sensitive computation offloading services in MEC. The reasons include NP-hardness of WDP, revenue deficiency, and difficulties handling the bidders' information when the auction is part of a larger sequence of commercial transactions [43].

To address the computational complexity of the WDP, I design an approximation algorithm that finds the allocation decisions in polynomial time. I notice that (4.17) is an instance of a multidimensional multiple-choice knapsack problem (MDMCKP) [20]. In that problem, a single knapsack consists of I containers/dimensions, each dimension with a resource constraint of R_i^n, and the knapsack packs exactly one task from each offloading UE in $s \in \mathcal{K}^n$ applying the "multiple-choice" constraint of MDMCK. It is well-known that MDMCKP is an NP-hard problem [21]. So, assuming a single dimension by relaxing the resource constraints in (C2), (4.17) boils down to an MCKP. Although MCKP is still NP-hard, it is solvable in pseudo-polynomial time using dynamic programming as long as the number of choices is low for each item [21].

Thus I use the knapsack dynamic programming-based resource allocations as the upper bound to the WDP (4.17). Afterward, I apply the VCG pricing rules to obtain a benchmark solution to the resource

allocation prices for computation offloading services. However, still, a more practical auction design to support dynamic resource allocations and the long sequences of service-oriented payment transactions for online auctions in MEC. Hence, I develop a computationally efficient and practically viable repeated auction model in the subsequent section, using the features from dynamic position auction and GSP mechanism [54].

4.3.3 GSP-Based Resource Allocation and Pricing Mechanism

In this section, I first outline a GSP-based allocation mechanism addressing the WDP (4.17). I summarize the modified GSP allocation and pricing algorithm in **Algorithm 3** that determines the offloading task assignment decisions, $x^n_{s,r_{im}}$, and corresponding allocation prices, p^n_s, given the tasks' priority scores, VMs' quality scores, and bids, as inputs.

For each processor n, I assume K distinct task positions following the position auction framework. Each position has a task priority index $\lambda^n_{s_j}$. So, the proposed mechanism first arranges the tasks in decreasing order of the requesting UEs' task priority indices as in $\lambda^n_{s_1} \geq \lambda^n_{s_2} \geq \cdots \geq \lambda^n_{s_K}$.

Besides, I consider the pool of VM resources, $\mathcal{R}^n$, as the list of items to allocate to the offloading task positions. I also define a function to rank the VMs according to their expected computation performance quality scores, θ^n_{i,r_m}, and the bids, b^n_{i,r_m}, submitted by their host server i. The ranking score of the m-th VM at server i is given by

$$y^n_{i,r_m} = \frac{\theta^n_{i,r_m}}{b^n_{i,r_m}}. \tag{4.18}$$

So, the proposed mechanism arranges the VMs into distinct rank positions as in: $y^n_{i,r_1} \geq y^n_{i,r_2} \geq \cdots \geq y^n_{i,r_{R^n}}$. Next, the mechanism sequentially matches the tasks and VMs according to their positions using the GSP auction allocation rules. That is, the task in the i-th position is matched to the VM in the i-th rank.

Besides, according to the GSP pricing rule, the corresponding allocation price is equal to the bid that the winning VM r^n_{im} requires to maintain its current m-th ranking position. Hence, the allocation price for the task position s, matched to the VM at rank $m = s$, satisfies

$$b^n_{i,r_s} \leq \frac{\theta^n_{i,r_s}}{\theta^n_{i,r_{s+1}}} b^n_{i,r_{s+1}}. \tag{4.19}$$

Thus, I define the price adjustment rate with respect to the VM in the m-th rank position as

$$\Theta^n_m = \frac{\theta^n_{i,r_m}}{\theta^n_{i,r_{m+1}}}. \tag{4.20}$$

The monotonicity in allocation prices means that a VM that wins a task in a higher position receives more than lower VMs. To satisfy that, I arrange the price adjustment rates in a decreasing order into $\mathbf{\Theta}^n_R$. Finally, the proposed modified GSP mechanism settles the allocation price for the s-th position as

$$p^n_s = \begin{cases} \Theta^n_{R_s} b^n_{i,r_{s+1}}, & \text{if } 1 \leq s < K \\ b^n_{i,r_s} + \epsilon, & \text{if } s = R^n \end{cases} \tag{4.21}$$

where ϵ is a small positive constant to guarantee that a VM is paid more than its bid, when it is the last one in the ranked list (i.e., $R^n = K$).

4.3.4 Repeated GSP-Based Dynamic Resource Allocation and Pricing Mechanism

In this section, I model the repeated GSP mechanism as a dynamic game of incomplete information. I consider a dynamic computation offloading environment, where the number of offloading tasks and the workloads of the VMs are uncertain until the offloading requests arrive at the processors.

Algorithm 3: *GSP_Mechanism*

Input: $\lambda^n, \theta^n, \boldsymbol{b}^n$
Output: $\pi^n = (\boldsymbol{x}^n, \boldsymbol{p}^n)$
Update the tasks' positions in the queue according to their priority indices, $\mathcal{K}^n \leftarrow sort(\lambda^n, \text{descend})$
Find each VM's ranking score,
for $m = 1$ **to** R^n **do**
  $y^n_{i,r_m} = \dfrac{\theta^n_{i,r_m}}{b^n_{i,r_m}}$
end
Arrange VMs according to their ranking scores, $\mathcal{R}^n \leftarrow sort(\boldsymbol{y}^n, \text{descend})$
Find the price adjustment rates,
for $m \leftarrow 1$ **to** $(R^n - 1)$ **do**
  $r_m \leftarrow \mathcal{R}^n[m]$, $r_{(m+1)} \leftarrow \mathcal{R}^n[m+1]$,
  $\Theta^n_m = \frac{\theta^n_{i,r_m}}{\theta^n_{i,r_{m+1}}}$
end
Arrange price adjustment rates in decreasing order, $\boldsymbol{\Theta}^n_R \leftarrow sort(\boldsymbol{\Theta}^n, \text{descend})$
Sequentially match the tasks in $\mathcal{K}^n$ with ranked VMs in $\mathcal{R}^n$, and find corresponding allocation prices.
for $s \leftarrow 1$ **to** K **do**
  if $(1 \leq s < K)$ **then**
    $x^n_{s,r_{is}} = 1$
    $p^n_s = \Theta^n_{R_s} b^n_{i,r_{s+1}}$
  end
  else if $(s == R^n)$ **then**
    $p^n_s = b^n_{i,r_s} + \epsilon$
  end
end

I summarize the repeated GSP-based resource allocation and pricing method in **Algorithm 4**. The algorithm begins with all the state information on UEs and VMs as inputs. At regular intervals with length T, the algorithm updates the priority indices for each processor n using (4.3) based on UEs' offloading historical data.

The algorithm continues the provisioning of offloading services at time $t = 1, 2, \ldots$ with a duration of Δt. At the beginning of each slot t, it gathers incoming offloading requests and places them into the queue $\mathcal{K}^n(t)$ according to the requested application type n. Any request that arrives after that will be processed in the next time slot. After queuing the offloading task in each processor $n \in \{1, 2, \ldots, N\}$, it initiates the bid collection process by sending a request for bids to the servers along with information on tasks' priority indices $\boldsymbol{\lambda}^n$ and resource allocation decisions from the previous round, i.e., $\pi^n(t-1)$. For initialization, it uses random task assignments.

Next, the algorithm updates the expected computation performance quality scores, $\theta^n_{i,r_m}(t)$, for the VMs based on their current workloads. Upon receiving the bids from all the servers, each processor runs **Algorithm 3** to obtain the allocation and pricing decisions $\pi^n(t)$ for the current round. Finally, the algorithm updates the workloads of the VMs according to the new allocation decisions before moving to the next auction round.

4.3.5 A Toy Example

Consider a MEC offloading scenario as illustrated in **Figure 4.3**, consisting of $J = 4$ offloading UEs, and $I = 2$ servers offering MEC services for $N = 3$ different applications. The MEC orchestrator gathers the

Algorithm 4: *Repeated_GSP_for_MEC_Offloading*

Input: $\mathcal{I}, \mathcal{J}, \mathcal{N}\}$
Output: $\{\pi^n(t)\}$

Initialize offloading service provisioning states,
$\mathcal{K}^n \leftarrow \emptyset, \mathcal{R}^n \leftarrow \emptyset, \mathcal{X}^n \leftarrow \emptyset, \mathcal{P}^n \leftarrow \emptyset$
Update task priority scores at every interval T.
for $n = 1$ **to** N **do**
  for $j = 1$ **to** K **do**
    $\lambda^n_{s_j} = \frac{\hat{\gamma}_{i',j}\beta^n}{\hat{d}^n_{s_j}\tau^n_{\max}f^n_{C_{\min}}}$
  end
end
Begin offloading service provisioning,
while $(t \in \{1, 2, \dots\})$ **do**
  Gather incoming offloading requests according to the requested task type,
  for $n = 1$ **to** N **do**
    for $s = 1$ **to** K **do**
      $K^n(t)[s] = d^n_{k_j}(t)$
    end
    Request servers to submit bids for this round,
    $\boldsymbol{b}^n(t) \leftarrow$ *Request_Bids* $(\lambda^n_{\text{sorted}}(t), \pi^n(t-1))$
    Update VM's computation quality scores,
    for $i = 1$ **to** I **do**
      for $m = 1$ **to** R^n_i **do**
        Find resource utilization metric $\phi^n_{i,r_m}(t)$ using eqn. (4.6)
        $\theta^n_{i,r_m}(t) = \frac{W^n_i f^n_{C_i}}{C^n_i}\phi^n_{i,r_m}$
      end
    end
    Obtain resource allocation and price decisions,
    $\pi^n(t) \leftarrow$ *GSP_Mechanism* $(\boldsymbol{\lambda}^n, \boldsymbol{\theta}^n(t), \mathbf{b}^n(t))$
    Update workloads for allocated VMs,
    for $i = 1$ **to** I **do**
      for $m = 1$ **to** R^n_i **do**
        for $s = 1$ **to** K **do**
          if $(x^n_{s,r_{im}}(t) == 1)$ **then**
            $\eta^n_{i,r_m}(t) = d^n_{k_j}(t)$
          end
        end
      end
    end
  end
end

incoming offloading requests and forwards them to the task processing queues. Each processor has a task queue with $K = 4$ distinct positions, where the task offloaded by UE j is placed in position $s = j$. For example, the tasks in processor $n = 3$ (i.e., $\boldsymbol{K}^3$) are listed in positions $s = 2$, $s = 3$, and $s = 4$ as per the UEs' indices.

The pool of VM resources represented by $\boldsymbol{R}^1$, $\boldsymbol{R}^2$, and $\boldsymbol{R}^3$, respectively in **Figure 4.3**, includes VMs from different servers. The VMs in processor $n = 1$ consists of $R_1^1 = 1$ and $R_2^1 = 3$ VMs from server $i = 1$ (shaded in grey color) and $i = 2$, respectively. Similarly, $\boldsymbol{R}^2$ consists of $R_1^2 = 3$ and $R_2^2 = 2$ VMs, and $\boldsymbol{R}^3$ has $R_1^3 = 2$ and $R_2^3 = 2$ VMs.

Consider the MEC system, with the set of offloading tasks and VM resources presented in **Figure 4.3**. I present an example describing the proposed GSP mechanism. The task priority indices of UEs are given as $\boldsymbol{\lambda}^1 = [0.31, 0.20, 0.15, 0.09]$, $\boldsymbol{\lambda}^2 = [0.13, 0.23, 0.14, 0.38]$, and $\boldsymbol{\lambda}^3 = [0.26, 0.11, 0.24, 0.20]$. Beside, **Figure 4.4** shows the VMs' quality scores and bids submitted for each application type.

Example 1 *Considering a MEC system, with the set of offloading tasks and VM resources as presented in Fig. 4.3, I hereby present an example describing how the proposed GSP mechanism determines the allocation decisions. I assume, the task priority indices of UEs are*

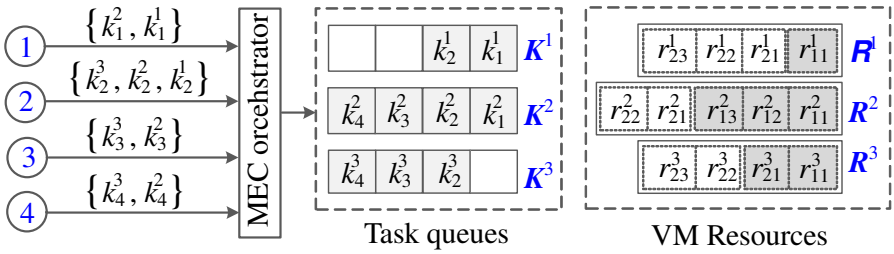
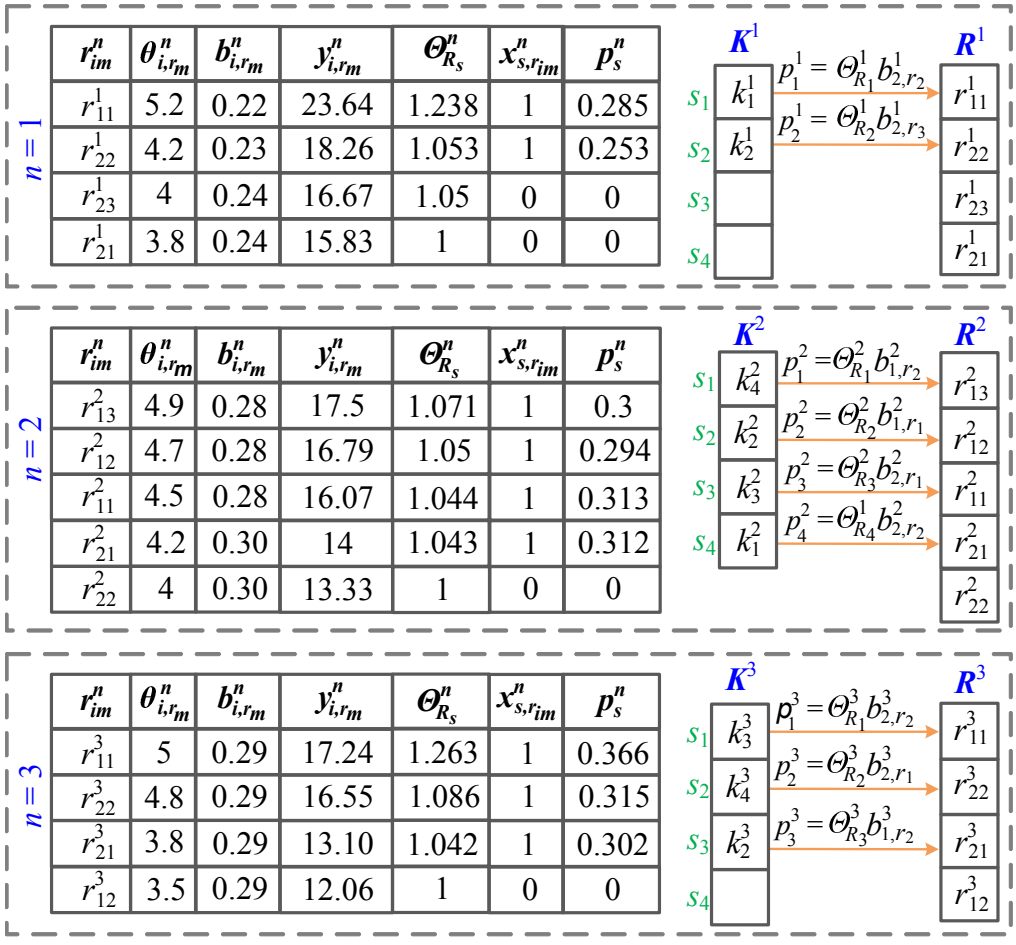


Figure 4.3: An illustration of incoming offloading requests and VMs at MEC processors.

given as: $\boldsymbol{\lambda}^1 = [0.31, 0.20, 0.15, 0.09]$, $\boldsymbol{\lambda}^2 = [0.13, 0.23, 0.14, 0.38]$, *and* $\boldsymbol{\lambda}^3 = [0.26, 0.11, 0.24, 0.20]$. *Given the VMs' quality scores and bids submitted for each application type, as in Fig. 4.4, the proposed mechanism for task assignment and pricing is described as follows.*

At first, the arriving tasks are sorted in decreasing order of their priority scores in their respective task queues. As demonstrated in **Figure 4.4)**, the tasks in $\mathcal{K}^1$ are arranged as $\lambda_{s_1^1} > \lambda_{s_2^1}$. Similarly, the tasks in $\mathcal{K}^2$ and $\mathcal{K}^3$ are arranged as $\lambda_{s_4^2} > \lambda_{s_2^2} > \lambda_{s_3^2} > \lambda_{s_1^2}$, and $\lambda_{s_3^3} > \lambda_{s_4^2} > \lambda_{s_2^3}$, respectively.



r_{im}^n	θ_{i,r_m}^n	b_{i,r_m}^n	y_{i,r_m}^n	$\Theta_{R_s}^n$	$x_{s,r_{im}}^n$	p_s^n
r_{11}^1	5.2	0.22	23.64	1.238	1	0.285
r_{22}^1	4.2	0.23	18.26	1.053	1	0.253
r_{23}^1	4	0.24	16.67	1.05	0	0
r_{21}^1	3.8	0.24	15.83	1	0	0

r_{im}^n	θ_{i,r_m}^n	b_{i,r_m}^n	y_{i,r_m}^n	$\Theta_{R_s}^n$	$x_{s,r_{im}}^n$	p_s^n
r_{13}^2	4.9	0.28	17.5	1.071	1	0.3
r_{12}^2	4.7	0.28	16.79	1.05	1	0.294
r_{11}^2	4.5	0.28	16.07	1.044	1	0.313
r_{21}^2	4.2	0.30	14	1.043	1	0.312
r_{22}^2	4	0.30	13.33	1	0	0

r_{im}^n	θ_{i,r_m}^n	b_{i,r_m}^n	y_{i,r_m}^n	$\Theta_{R_s}^n$	$x_{s,r_{im}}^n$	p_s^n
r_{11}^3	5	0.29	17.24	1.263	1	0.366
r_{22}^3	4.8	0.29	16.55	1.086	1	0.315
r_{21}^3	3.8	0.29	13.10	1.042	1	0.302
r_{12}^3	3.5	0.29	12.06	1	0	0

Figure 4.4: An example demonstrating the assignment of offloading tasks to VMs, and corresponding GSP-based pricing mechanism.

Next, the VMs in each processor are arranged according to their ranking scores. VMs in $\mathcal{R}^1$ are ranked as $y^1_{1,r_1} > y^1_{2,r_2} > y^1_{2,r_3} > y^1_{2,r_1}$. Similarly, other VMs are ranked as $y^2_{1,r_3} > y^2_{1,r_2} > y^2_{1,r_1} > y^2_{2,r_1} > y^2_{2,r_2}$ in $\mathcal{R}_2$. Finally, I have $y^3_{1,r_1} > y^3_{2,r_2} > y^3_{2,r_1} > y^3_{1,r_2}$ in $\mathcal{R}_3$.

Now, the GSP resource allocation and pricing mechanism (**Algorithm 3**) sequentially matches the tasks in each queue $\mathcal{K}^n$ to the VMs in the queue $\mathcal{R}^n$, as shown by directed arrows in **Figure 4.4**. For example, task k^1_1 in the first position in $\mathcal{K}^1$ is matched to VM r^1_{11}, which is in the top rank in $\mathcal{R}^1$. The corresponding task assignment decision variable is then updated as $x^1_{1,r_{11}} = 1$.

The allocation prices are then determined based on the price adjustment rates and the bid of the VM ranked next. For example, the allocation price for the first task position yields $p^1_1 = \Theta^1_{R_1} b^1_{2,r_2} = (1.238 \times 0.22) = \$0.285$/VM-hr. The same procedure follows for all other tasks in the processors.

4.4 Analysis of Bidding Strategies in GSP-Based Dynamic MEC Offloading Auction

In this section, I study the strategic behavior of MEC servers. The servers who participate as bidders can adjust their bids in every time slot. They can learn about competitors' bids through interactions over successive auction rounds and adjust their strategies to maximize their utility. To that end, I study best-response bidding strategies for servers to adjust bids within a certain range. That also guarantees to achieve a Nash equilibrium in every round, which maximizes the utility for every participating server.

4.4.1 Adaptive Balanced Bidding Strategies of Servers

I consider myopic best-response strategies [8], where every server i adjusts its bids for the current auction round t under the assumption that other servers repeat their bids of the previous round. In the dynamic computation offloading setting, the CPU utilization and the computation performance qualities of VMs fluctuate based on their computational workloads. Hence, I consider an adaptive balanced bidding policy, where each server devises the current bidding strategies considering the VMs' expected computation performance quality scores and adjusts the bids accordingly.

In the position auction framework, the offloading tasks in the higher positions return higher profits for VMs, due to the monotonicity in GSP prices. Hence, every bidder adjusts its bids for each VM by targeting a higher task position that can maximize the VM's profit in the next auction round. To win a task with a higher offloading rate, a lower value is necessary so that the VM can obtain a higher-ranking position in the auction. On the contrary, self-interested bidders tend to bid higher for their VMs with higher expected quality scores to earn more profits.

Therefore, I propose a balanced bidding strategy that allows a server to adjust the bids for its VMs within a certain range, and yet maximize the profit. To restrict servers from overbidding, I use the restricted balanced bidding (RBB) strategy [10]: A server determines the bid for each of its VMs by aiming for the desired task position s_m^* that maximizes the m-th VM's utility. It then adjusts its bid based on the allocation price of the target slot $p^n_{s_m^*}(t-1)$ in the previous round.

The core concept of balanced bidding is limiting the degree to which a server acts greedy while adjusting the bids for VMs. In the RBB strategy, when a server adapts its bid for each VM r_{im}^n, it can only aim for task positions with no higher priority indices than the position that the m-th VM has already won in the previous round. That allows the server to bid as low as essential to secure the s_m^*-th rank in the next auction round. Moreover, it adjusts the bid in a balanced way so that if the VM cannot win the target slot, it does not end up with a profit lower than the previous round. Below, I define the RBB strategy formally.

Definition 6 *Assuming other servers' bids remain fixed at their previous values, and $\hat{s}_m$ be the position that the m-th VM at server i has won in the previous round $(t-1)$, the* ***Restricted Balanced Bidding (RBB)*** *strategy is to:*

(i) find the target task position s_m^* among the positions ranging from $\hat{s}_m$ to K, that maximizes the VM's utility:

$$s_m^* = \arg\max_{s'} \left\{ \lambda_{s'_j}^n \theta_{i,r_m}^n(t) \left(p_{s'}^n(t-1) - v_{i,r_{s'}}^n \right), \right\},$$

If the VM has not been allocated to any position is the previous auction round, then the server looks for the target position within the range: $1 \leq s' \leq K$.

(ii) adjust the bid for the current auction round t, in a way that satisfies the following

$$\begin{aligned} &\lambda_{s_m^*}^n \theta_{i,r_m}^n(t-1) \left(p_{s_m^*}^n(t-1) - v_{i,r_m}^n \right) \\ &\quad = \lambda_{s_m^*-1}^n \theta_{i,r_m}^n(t) \left(b_{i,r_m}^n(t) - v_{i,r_m}^n \right). \end{aligned}$$

Algorithm 5: *Restricted_Balanced_Bidding_Strategy*

Input: λ^n, $\boldsymbol{x}^n(t-1)$, $\boldsymbol{p}^n(t-1)$

Output: $\boldsymbol{b}_i^n(t)$

Upon receiving bid request from processor n,
for $m = 1$ **to** R_i^n **do**
  Find whether the VM has won any task position in the previous time slot $(t-1)$,
  for $s = 1$ **to** K **do**
    if $(x^n_{s,r_{im}}(t-1) == 1)$ **then**
      $\hat{s}_m = s$
      else
        $\hat{s}_m = 1$
      end
    end
  end
  Find the target slot s_m^* that maximizes utility,
  for $s' = \hat{s}_m$ **to** K **do**
    $\hat{u}^n_{s',m} = \lambda^n_{s'_j}\theta^n_{i,r_m}(t)\left(p^n_{s'}(t-1) - v^n_{i,r_{s'}}\right)$
  end
  $s_m^* \leftarrow \arg\max(\hat{\boldsymbol{u}}^n)$
  Adjust bid according to RBB strategy in eqn. (4.22),
  if $(s_m^* == 1)$ **then**
    $\Pi^n_{i,r_m} = \dfrac{\theta^n_{i,r_m}(t-1)}{2\,\theta^n_{i,r_m}(t)}$
  end
  else
    $\Pi^n_{i,r_m} = \dfrac{\lambda^n_{s_m^*}\,\theta^n_{i,r_m}(t-1)}{\lambda^n_{s_m^*-1}\theta^n_{i,r_m}(t)}$
  end
  $b^n_{i,r_m}(t) \leftarrow v^n_{i,r_m} + \Pi^n_{i,r_m}\left(p^n_{s_m^*}(t-1) - v^n_{i,r_m}\right)$
end

Using the RBB strategy as defined above, the updated bid for the current auction round t for the m-th VM at server i yields

$$b^n_{i,r_m}(t) = v^n_{i,r_m} + \Pi^n_{i,r_m}\left(p^n_{s_m^*}(t-1) - v^n_{i,r_m}\right) \tag{4.22}$$

where $\Pi^n_{i,r_m} = \dfrac{\lambda^n_{s_m^*}\,\theta^n_{i,r_m}(t-1)}{\lambda^n_{s_m^*-1}\theta^n_{i,r_m}(t)}$. If the target task position is the topmost one, i.e., $s_m^* = 1$, then I assume $\lambda_0^n = 2\lambda_1^n$ to adjust the bid. In **Al-**

gorithm 5, I describe adapting the bid by a server following the RBB bidding strategy for the VMs in every processor n.

4.4.2 Analysis of Bidding Dynamics on Auction Efficiency

In this section, I analyze the efficiency of the GSP mechanism under dynamic MEC offloading setting considering synchronous bidding model [54], where servers update their bids simultaneously. I show that no server encounters a negative utility gain through their participation in the auction satisfying the *Individual Rationality (IR)* property, when each server follows the RBB strategy in every round of auction (**Theorem 1**). First, I define the Individual Rationality (IR) property.

Definition 7 *In the GSP-based MEC offloading mechanism, the* ***Individual Rationality (IR)*** *is satisfied, if the resource allocation prices* $p_s^n(t)$ *guarantee non-negative utility gain, i.e.,* $u_{i,r_m}^n(t) \geq 0$ *for every VM* r_{im}^n *at server* i *that participates in the computation procedure at processor* n *during the time slot* t*.*

Theorem 1 *The GSP-based MEC offloading auction at each processor* n *guarantees the individual rationality for every participating VM, when the servers follow the proposed RBB strategy in every round of auction.*

Proof. See Appendix B.1. ■

Next, I analyze the stability of the GSP-based resource allocation outcomes in a dynamic setting, the same auction mechanism is repeated in every time slot t with new sets of offloading requests at N processors. In this scenario, the concept of stability in the auction mechanism in each processor n is represented in terms of an equilibrium point with

a set allocation price decisions, where every server is well-off with the resource allocation decisions and do not wish to exchange any of the VM allocation with another VM during the time slot t. Therefore, the auction reaches N distinct equilibrium points at different processors in every time slot t. In Theorem 2, I show that the proposed GSP-based MEC offloading mechanism results into a set of *Symmetric Nash Equilibrium (SNE)* allocation prices in every auction round, where no server prefers to exchange the assigned task positions for any of its VM, with another task position within the same processor. Instead the servers are well-off with their utilities at SNE, and thus maintain the equilibrium by following the RBB strategy for the future auction round. I formally define the SNE allocation prices in a dynamic MEC offloading scenario, as follows:

Definition 8 *The set of resource allocation prices $\boldsymbol{p}^n(t)$ at processor n during the time slot t, is in **Symmetric Nash Equilibrium (SNE)** if the following holds for any task positions s and s' in $\mathcal{K}^n(t)$:*

$$\lambda^n_{s_j}\left(p^n_s(t) - v^n_{i,r_s}\right) \geq \lambda^n_{s'_j}\left(p^n_{s'}(t) - v^n_{i,r_{s'}}\right), \tag{4.23}$$

Theorem 2 *There exists a set of symmetric Nash equilibrium (SNE) allocation prices for GSP-based MEC offloading auction at each processor n, when every server i follows the proposed RBB strategy in every round of auction.*

Proof. See Appendix B.2. ■

In the dynamic setting, the auction has a set of SNEs and thus may reach different SNE points at various auction rounds. However, the bidders can vary their bids within a certain range without violating the

stability of the allocation outcomes. Theorem 3 presents the upper and lower bound to the SNE for the proposed GSP-based resource allocation mechanism.

Theorem 3 *For any task type n, the upper- and lower bounds for the bid to satisfy the SNE conditions are given by*

$$b_{i,r_s}^{n,\mathrm{UB}} x_{(s-1),r_{(s-1)}}^{n} = \frac{v_{i,r_s}^{n}}{\Theta_{R_{(s-1)}}^{n}} x_{(s-1),r_{(s-1)}}^{n} + \frac{\lambda^{*}\Theta_{R_s}^{n}}{\Theta_{R_{(s-1)}}^{n}} \left(b_{i,r_{(s+1)}}^{n} - v_{i,r_s}^{n} \right) x_{s,r_s}^{n}, \tag{4.24}$$

$$b_{i,r_s}^{n,\mathrm{LB}} x_{(s-1),r_{(s-1)}}^{n} = \frac{v_{i,r_{(s-1)}}^{n}}{\Theta_{R_{(s-1)}}^{n}} x_{(s-1),r_{(s-1)}}^{n} + \frac{\lambda^{*}\Theta_{R_s}^{n}}{\Theta_{R_{(s-1)}}^{n}} \left(b_{i,r_{(s+1)}}^{n} - v_{i,r_{(s-1)}}^{n} \right) x_{s,r_s}. \tag{4.25}$$

where $\lambda* = \frac{\lambda_{s_j}^{n}}{\lambda_{(s-1)_j}^{n}}$, $r_{(s-1)}$, r_s, *and* $r_{(s+1)}$ *denotes the VMs allocated to the task positions* $(s-1)$, s, *and* $(s+1)$, *respectively.*

Proof. See Appendix B.3. ■

To validate that the proposed GSP-based MEC offloading auction reaches the equilibrium point within a polynomial time in every auction round, I analyze the computational complexity of the proposed resource allocation and pricing algorithms. Using **Algorithm 4** in every time slot t, one can obtain the computation offloading task assignment and allocate VM resources. The algorithm has prior knowledge about the task priority indices. Theorem 4 states some results about the computational efficiency of our proposed mechanism.

Theorem 4 *The proposed GSP-based MEC offloading auction is computationally efficient, i.e., the resource allocation outcomes are determined in polynomial time in every auction round t.*

Proof. See Appendix B.4 ■

4.5 Numerical Results

In this section, I first study the convergence properties of the proposed RBB bidding strategy considering the static scenario of computation offloading. Next, I compare the auction's revenue properties of the proposed GSP mechanism with the well-known welfare-maximizing VCG mechanism. I also investigate the mechanism's performance from the users' and overall MEC system's perspectives concerning welfare maximization.

Table 4.2: System parameters used for simulation in Chapter 4

Parameters and values	
$N = 1$, $I = 2$	$J = 150$, $K = J\,\Delta_t = 1$ min
$f_{C_{\min}} = 3.2$ GHz	F_t= 5.8 GHz
$[\tau_{\min}, \tau_{\max}] = [20, 200]$ msec	$P_u = 20$ dBm
BW = 80 MHz	σ_N^2 = -100 dBm
$[\Gamma_{\min}, \Gamma_{\max}] = [0.01, 0.90]$	$\mu_d = 2.12, \mu_0 = 29.2, \mu_f = 2.11$
$d_k \sim$ Poisson(d_{avg}) MB	$d_{\text{avg}} \sim [10, 40]$ MB
$\kappa = 10^{-24}, \epsilon = 0.001$	$q_c = 0.5, q_l = 0.5$
$\bar{a} =$ \$ 20/month	

I consider a dynamic simulation setup for the MEC system that manages computation offloading service provisioning within a $250 \times 250\text{m}^2$

Table 4.3: VM Configuration Model used for simulation in Chapter 4

MEC Server i	R_i^n	W_i	RAM (GB)	$f_{C_i}^n$ (GHz)	C_i (MIPS)	ρ_i ($/VM-hr)
$i = 1$	60	2	4	3.3 GHz	32	0.0452
$i = 2$	60	2	4	3.5 GHz	24	0.0435
$i = 3$	40	2	4	3.2 GHz	24	0.0385
$i = 4$	50	1	2	3.2 GHz	16	0.0186
$i = 5$	50	1	2	3.3 GHz	16	0.0175

area. There are $I = 5$ servers/nodes and $J = 150$ wireless UEs, randomly located within this area following uniform and non-uniform distribution, respectively. The UEs are associated with the nearest physical MEC node. The offloading requests are generated randomly in each auction round, where the offloading data sizes follow the Poisson distribution with the parameter d_{avg}. **Table 4.2** lists the MEC offloading and wireless channel parameters. **Table 4.3** provides the VM configuration model. Each server offers a single type of VM instance compatible with the $n = 1$-th MEC application. To capture the dynamics of the wireless channel and offloading process more accurately, I simulate 1000 times for each time slot and use the average for that auction round.

4.5.1 Convergence of Bidding Strategies

To analyze the convergence properties for the proposed GSP-based mechanism, I consider a static offloading scenario assuming all $J = 150$ UEs offload the same length of computing tasks in every round of auction. Furthermore, there are $I = 2$ servers with private valuations $v_1 = \$\,0.0354$ and $v_2 = \$\,0.0366$, respectively. I study their competitive

bidding behavior by following the proposed RBB strategy. I investigate three cases with the various number of available VMs, i.e., $R = [150, 1]$, $R = [1, 150]$, and $R = [80, 80]$, representing different levels of competition among the servers. The first two cases represent the monopoly market scenario where server $i = 1$ and server $i = 2$ dominate the supply of VM resources, respectively. The third case corresponds to fair competition among servers. To identify the convergence point, I set the convergence tolerance to 0.0001.

Figure 4.5 shows the average bids of the servers and the respective allocation prices according to the submitted bids by each server for the above three cases. The solid- and dashed lines represent the bids submitted by the server $i = 1$ and the server $i = 2$, respectively. In all three cases, the servers bid no less than their private valuations (as shown by the green lines in **Figure 4.5(a)**), ensuring non-negative profit/utility gain. Every server tends to increase its bid prices to beat the opponent, which gradually converges to some fixed points after a finite number of auction rounds. **Figure 4.5(b)** shows the corresponding allocation prices, where I compare the proposed GSP-based allocation prices with the VCG-based prices obtained from the Knapsack-based dynamic programming solution approach for (4.17). In all three cases, the VCG-based allocation prices are constant and lower than the proposed GSP-based prices, confirming that VCG mechanism is welfare-maximizing (i.e., buyer-friendly).

In the first case, the bids submitted by server $i = 1$ and server $i = 2$ converge to $b_1^* = \$\,0.0371$/VM-hr in time slot $t = 11$ and $b_2^* = \$\,0.0384$/VM-hr in time slot $t = 6$, respectively (**Figure 4.5(a)**).

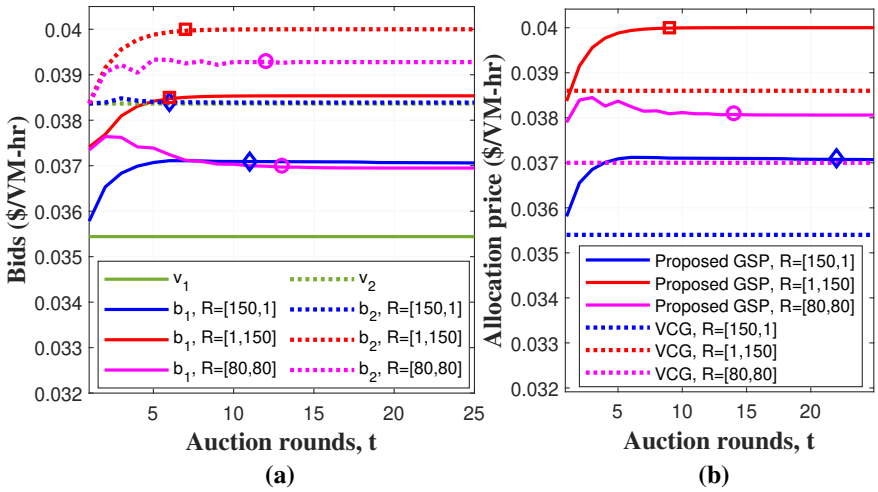


Figure 4.5: Convergence analysis for the proposed RBB bidding strategies, based on (a) average bid prices of each server, b_i^n, and (b) average allocation prices, p^n, considering $I = 2$ servers with varying number of VMs, R_i^n.

In this case, server $i = 1$ dominates the market supply. Thus the majority of the offloading tasks go to VMs at that server. The corresponding allocation prices reach the equilibrium point $p^* = \$\,0.0371$/VM-hr in round $t = 22$ (**Figure 4.5(b)**). There is a similar trend for the second case. When server $i = 2$ dominates the market supply, it wins the majority of the tasks. Server $i = 1$'s bid converges to $b_1^*(t = 6) = \$\,0.0385$/VM-hr, which is significantly higher than the previous case indicating that the server tries to beat the opponent despite its lower supply of VM resources. The bid prices of the other server $i = 2$ with a higher supply of VM resources converge to $b_2^*(t = 7) = \$\,0.04$/VM-hr, which indicates the server's influence over the allocation prices converging to the equilibrium point $p^*(t = 9) = \$0.04$. When the market has a fair level of competition as in the third case, both servers' bids converge to lower values, i.e., $b_1^*(t = 13) = \$0.0370$/VM-hr and $b_2^*(t = 13) = \$0.0393$/VM-hr, respectively. The corresponding allocation prices, shown in **Figure 4.5(b)**, confirm that bringing more sellers

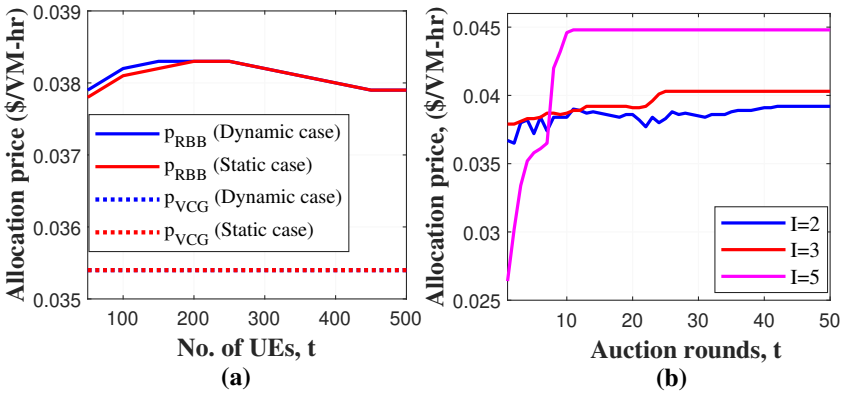


Figure 4.6: Comparison of average allocation prices for (a) varying number of UEs, J; and (b) different number of MEC servers, I.

into the offloading service market influences the market to settle at a lower equilibrium price, $p^*(t = 14) = \$0.0381$/VM-hr, which is more favorable to the UEs.

Next, I analyze the convergence of the proposed RBB strategies considering the dynamic offloading scenario, where the size of the computing tasks changes following the Poisson distribution. In **Figure 4.6(a)**, I compare the average allocation prices for the proposed GSP mechanism with the VCG mechanism for varying numbers of UEs, starting from $J = 50$ to $J = 500$. Furthermore, I assume the size of the offloading task queue is the same as the number of UEs, i.e., $K = J$, and the number of VMs at the servers as $\boldsymbol{R} = [250, 250]$. As shown in **Figure 4.6(a)**, the average allocation prices for both the static and dynamic cases are very close for a lower number of UEs. Eventually, as the number of UEs increases, they coincide. In the beginning, with $J = 50$, the resource utilization is low. Hence, the proposed GSP mechanism selects low allocation prices, i.e., $p_{\text{GSP}} = \$\,0.0378$/VM-hr and $p_{\text{GSP}} = \$\,0.0379$/VM-hr for the static and dynamic cases, respectively. As the number of UEs, thus resource utilization and competition among the server, increases, the corresponding allocation prices increase. They

reach the maximum of $p_{\text{GSP}} = \$\,0.0383$/VM-hr for both the static and dynamic cases at $J = 250$. At this point, the servers exhibit the maximum level of competition since, for each UE, the probability of being matched with either server is equal. When the number of UEs grows to even larger values, the allocation prices start to drop because the increasing workloads at the VMs restrict the servers from raising the bids further. In both the static and dynamic cases, the VCG-based allocation prices remain fixed at $p_{\text{VCG}} = \$\,0.0354$/VM-hr, lower than the proposed GSP-based prices.

In **Figure 4.6(b)**, I compare average allocation prices for the proposed GSP mechanism for different numbers of resource sellers in the offloading service market. Considering a dynamic offloading scenario with $J = 120$ UEs and a total of $R = 120$ VMs, I study three different cases with $I = 2$, $I = 3$, and $I = 5$, each server having $R_i = 60$, $R_i = 40$, and $R_i = 24$ VMs, in each case respectively. As the number of servers increases, the competition among the resource sellers grows, which increases the average allocation prices. When the number of servers $I = 5$, the average allocation prices converge to $p^* = \$\,0.0448$/VM-hr which is approximately 11%, and 14% higher than the cases with $I = 2$ and $I = 3$, respectively.

4.5.2 Auction Revenue and Profits of Servers

In this section, I evaluate the auction performance of the proposed GSP-based offloading mechanism for the proposed RBB strategy and compare the results with the other bidding strategies. For comparison, I consider the greedy bidding strategies in GSP mechanism [9], which include: (i) balanced bidding (BB) strategy, where servers can target

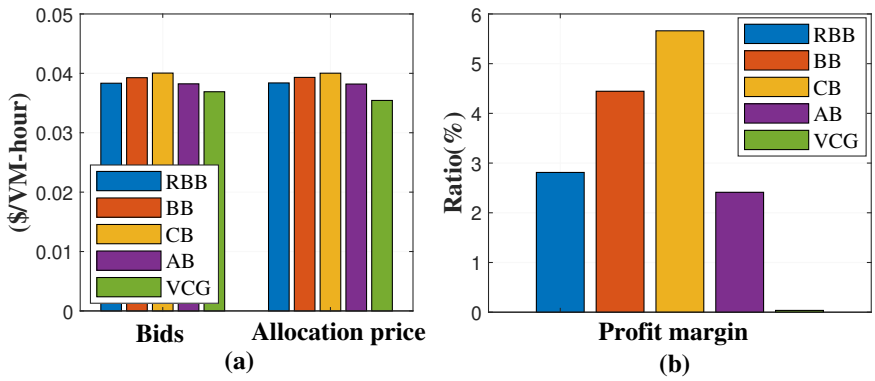


Figure 4.7: Performance comparison between knapsack-based VCG mechanism and GSP mechanism with different balanced bidding strategies, based on: (a) Submitted bids vs allocation prices; (b) Profit margin ratio (%) of servers.

any task position maximizes its utility gain, (ii) altruistic bidding (AB) strategy, where servers always bid lower than the allocation price of the target task position favoring the buyers, and (iii) competitor busting (CB) strategy, where the servers act vindictively by bidding higher than the allocation price of the target task position so that the competitor ends up with a lower utility gain.

Considering $I = 2$ servers with $R = [80, 80]$ VMs and $J = 150$ UEs, I first compare the equilibrium bids (i.e., the point of bid convergence) and allocation prices of the proposed strategy with other greedy strategies and truthful bidding in VCG mechanism. As shown in **Figure 4.7(a)**, the CB strategy results in the highest average bid, thus the highest allocation price; nevertheless, it is not suitable for both sellers and buyers. Among other strategies, the RBB strategy gives higher bids and allocation prices with $b^*_{\text{BB}} = p^*_{\text{BB}} = \$\,0.0393$/VM-hr, whereas our proposed BB strategy converges to slightly lower values, i.e., $b^*_{\text{RBB}} = \$\,0.0383$/VM-hr and $p^*_{\text{RBB}} = \$0.0384$/VM-hr. That is because the BB strategy allows servers to choose their preferable target positions without any restriction, unlike our proposed strategy. The

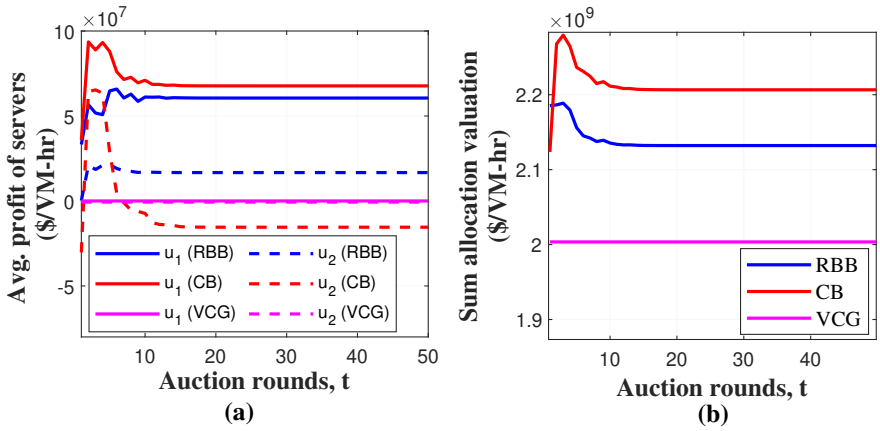


Figure 4.8: Performance evaluation of proposed RBB strategy in comparison with other bidding strategies, based on (a) average profits of servers; and (b) sum allocation valuation

BB strategy also performs well in terms of a profit margin ratio of 4.44% (**Figure 4.7(b)**). However, our proposed RBB strategy results in 2.811%, which is still better than the AB strategy and the VCG mechanism.

Next, I analyze the average profit of each server and the auction revenue, i.e., the sum valuation of task computation at the allocated resources. I compare the results for the proposed RBB strategy with the truthful bidding in the VCG mechanism and the CB strategy, representing the best-case and the worst-case scenarios from the users' perspectives. As shown in **Figure 4.8(a)**, the average profits of server $i = 1$ and $i = 2$ for the proposed RBB strategy converge to $u_i^* = \$\,6.07 \times 10^7$/VM-hr and $u_i^* = \$\,1.67 \times 10^7$/VM-hr, respectively. For the CB strategy, although the average profit of server $i = 1$ becomes even higher than the proposed RBB strategy (shown with the solid red line), server $i = 2$ ends up with a negative utility gain because of the vindictive bidding behavior. The servers' gains for the VCG mechanism remain zero. A similar trend appears in **Figure 4.8(b)**, where

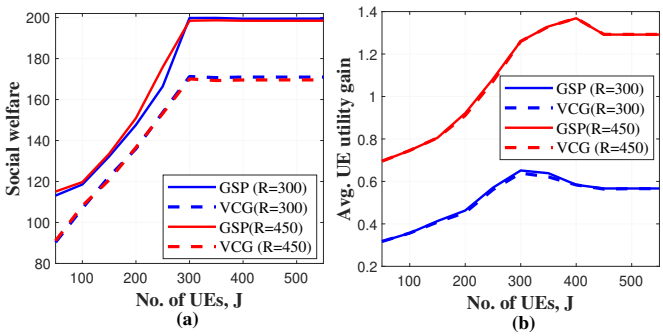


Figure 4.9: Performance comparison between proposed GSP and VCG mechanism for varying number of VMs, based on: (a) Social welfare; (b) Average utility gain of UEs.

the proposed RBB strategy yields a higher sum allocation valuation than the VCG mechanism, whereas the CB strategy exceeds the total valuation due to higher allocation prices.

4.5.3 Social Welfare and QoE Analysis

In this section, I analyze the performance of the proposed GSP-based offloading mechanism addressing the QoE of UEs and the overall social welfare of the MEC system. With $I = 3$ servers and $K = 300$, I compare the social welfare and the average utility of UEs of the proposed GSP mechanism with the VCG mechanism in **Figure 4.9**. To that end, I change the number of UEs from $J = 50$ to $J = 550$. I evaluate the results for two cases, $R = 300$ and $R = 450$, with $R_i = 100$ VMs and $R_i = 150$ VMs in each server i, respectively. **Figure 4.9(a)** shows that the social welfare for the GSP- and VCG mechanisms increases as the number of UEs increases. It then remains fixed after the cut-off point $J = K = 300$ when the number of UEs surpasses the capacity limit. Furthermore, the proposed GSP mechanism provides higher social welfare than the VCG mechanism for both cases. Although the VCG mechanism guarantees higher utility gain for the UEs than the

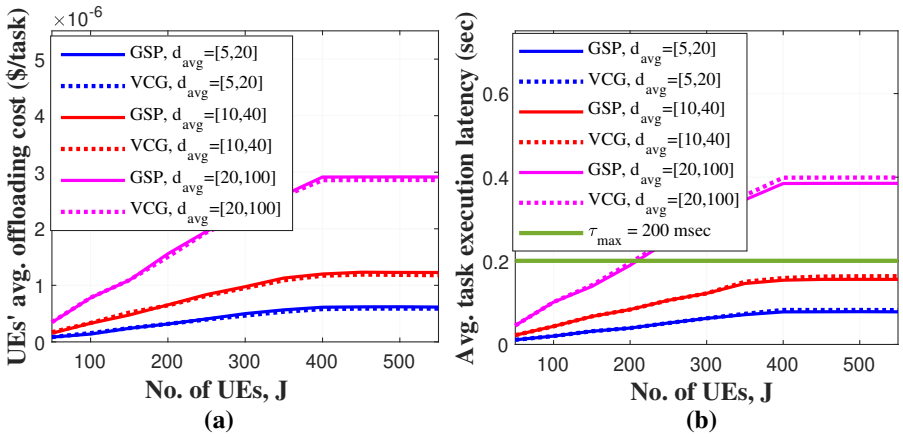


Figure 4.10: QoE Analysis for the proposed GSP and VCG mechanisms for varying offloading task sizes, based on: (a) average offloading cost; (b) average offloading service latency.

proposed GSP mechanism (**Figure 4.9(b)**), the latter results in higher social welfare than the former because the profit of the servers is always higher. **Figure 4.9(b)** also reveals that the average utility gain of UEs for the GSP- and VCG mechanisms coincide for both cases. When the MEC system has the same resource availability as the system capacity limit, i.e., $R = 300$, then the average utility gain increases until the number of UEs hits the system capacity, i.e., $K = 300$. It then decreases until it reaches the fixed point at $Q^* = 0.567$. For the second case with $R = 450$, the average utility gain of UEs improves with more supply of VM resources for both mechanisms. In this case, the UEs' average QoE reaches its maximum when $J = 400$ and then goes down to the fixed point of $Q^* = 1.291$. In the case of social welfare, the changes in VM resource supply (e.g., $R = 300$ and $R = 450$) have no significant impact on the mechanisms (**Figure 4.9(b)**).

Next, I study the QoE of UEs for three different cases by varying the average offloading task sizes: (i) $d_{\text{avg}} = [5, 20]$, (ii) $d_{\text{avg}} = [10, 40]$, and (iii) $d_{\text{avg}} = [20, 100]$. With $R = 450$ and $K = 400$, I plot the average

offloading cost and service latency of UEs in **Figure 4.10**. The results are almost the same for both mechanisms. The average offloading cost and service latency of UEs tend to increase as the number of UEs increases until it reaches the system's capacity limit, i.e., $K = 400$, and then remain fixed. Moreover, when the average offloading data sizes increase, the corresponding average offloading cost and service latency of UEs rise significantly. To be specific, when the average offloading task size is $d_{\text{avg}} = [20, 100]$, the average offloading service latency exceeds the maximum task completion threshold, $\tau_{\max} = 200$ msec, after the number of UEs reaches $J = 200$. The average offloading cost for the case with $d_{\text{avg}} = [20, 100]$ is more than double the offloading compared to others.

4.6 Conclusion

In this chapter, I presented a dynamic resource allocation mechanism for MEC offloading, which implements computation offloading as a service via a repeated GSP auction process. The main objective of the proposed mechanism is to maximize the sum valuation of the computing resources at the servers while satisfying the QoE of offloading users. Furthermore, I proposed a restricted balanced bidding (RBB) strategy for the servers, which guarantees the symmetric Nash equilibrium (SNE) condition for the proposed GSP mechanism in every round of auction. To validate the performance of the proposed solution approach, numerical results are given along with theoretical analysis.

Chapter 5

Conclusion and Future Research Direction

In this chapter, I briefly summarize the research contributions and conclude the thesis. I also discuss about the open problems in MEC offloading, which can be addressed using repeated GSP auction framework in future research.

5.1 Conclusion

This thesis introduces a novel application of GSP-based position auction into MEC-enabled wireless networks, specifically for deploying computation offloading services for 5G networks and beyond. I presented a generic business model to implement service-oriented MEC offloading in the presence of multiple competitive computation service providers. This MEC service delivery pltaform implements computation offloading as a service trading betweeen wireless UEs and MEC servers through an online auction process, which centrally manages the overall resource allocation and service pricing mechanisms.

Furthermore, I investigate efficient resource allocation and pricing mechanism design approaches, considering the realistic network dy-

namics and the challenges related to satisfying hetereogeneous QoS requirements of users in wireless environment. The main research objective of this thesis is to design an efficient auction mechanism, so that the limited computing resources are allocated in a way that satisfies offloading users' QoS demands and also ensures profit gain for MEC service providers.

Towards achieving this goal, I design a novel GSP-based reverse auction model in Chapter 3, which efficiently allocates computing resources satisfying users' resource demands and also maximizes the profits of MEC servers. To guarantee the stability and allocation efficiency of the proposed resource allocation mechanism, I propose adaptive balanced bidding strategies for MEC servers so they cannot overbid and manipulate the resource pricing mechanisms in a dynamic offloading environment. Numerical results are presented to evaluate the performance of the proposed mechanism from both the users' and servers' perspectives.

I perform further investigation on enhancing users' QoE in a dynamic wireless environment, addressing the challenges in allocating resources when offloading requests arrival at random intervals, and computational workloads and corresponding CPU performance of the servers vary in each offloading period. I develop effective dynamic resource management policies using the concepts of priority queuing and network function virtualization approaches, and present a flexible MEC offloading framework in Chapter 4. This MEC service delivery platform can be integrated into the existing wireless network architecture, to deploy MEC services for various applications using SDN function-

alities. A detailed orchestration workflow is presented to model the communication and offloading service provisioning process.

To implement this MEC service provisioning process, I design a repeated GSP auction mechanism addressing the competition among the computing resource sellers. The dynamic resource allocation problem in the auction procedure involves the offloading task assignment decision problem (i.e., WDP) and offloading service pricing decision problems. I mathematically formulate the WDP as an optimization problem, that maximizes the total allocation valuation for different types of MEC application under the resource capacity constraints of the servers. I present new load-ware ranking and pricing rules to the develop the GSP-based resource allocation and pricing algorithm. Moreover, to enforce regulation over the competitive bidding behavior of the MEC servers, I propose new restricted-balanced bidding strategies so that the auction results stable and economically efficient allocation outcomes in every offloading period. A through analysis of the bidding strategies and auction properties is presented. An extensive numerical analysis is also given showing the performance of the proposed mechanism in achieving social welfare and allocation efficiency in comparison with the existing approaches.

5.2 Future Research Directions

In the next generation of MEC networks, the efficient management of limited resources of MEC servers is continued to be the fundamental problem, as the number of wireless traffic increases exponentially. Therefore, when multiple computation service providers enter into the

MEC service market, the resource allocation problem turns into multi-fold, such as, satisfying heterogeneous QoS demands at minimal offloading cost, and handling the competition among the service providers. Furthermore, the trade-off between the offloading cost and QoS always exist in the MEC offloading scenario.

Related to the proposed repeated GSP auction-based MEC offloading mechanisms presented in Chapter 3 and Chapter 4 of this thesis, I outline few potential enhancements and extensions as the future research:

- In the multi-user multi-vendor MEC offloading scenario, it is crucial to preserve the users' private information, e.g., physical locations, the network operator/ service provider they belong to, payment information, etc. In order to protect users from cyber attacks, block-chain technology can be integrated with the MEC offloading architecture. Similarly, the block-chain enabled security can also be enforced for MEC servers for a regulated interaction during the bidding process.

- In order to accurately model the uncertainties in users' QoS requirements, mathematical tools such as Markov decision processes and reinforcement learning can be applied. This will help model the randomness in offloading task arrivals and users' mobility, then efficiently allocate the resources as the users' offloading status evolve.

- To ensure fairness and effectiveness considering the dynamic priority queuing of the offloading tasks and corresponding load-aware

computation resource management policies, stochastic optimization approaches can be investigated with the goal of maximizing social welfare of the offloading system.

- In addition, the optimal bidding strategies for the competitive MEC servers can be studied, using the probabilistic mathematical models to predict the resource utilization and expected profit gain, and then adjust bid prices with revenue guarantee.

- As the truthful bidding is not the dominant strategy in GSP auction, it is difficult to identify vindictive bidding and prevent bidders from trying to manipulate the allocation prices. So, to ensure payment fairness and price stability, rational pricing strategies need to be devised correcting the utility loss caused by vindictive bidding.

Bibliography

[1] Gagan Aggarwal, Ashish Goel, and Rajeev Motwani. "Truthful Auctions for Pricing Search Keywords". In: *InProceedings of 7th ACM Conference on Electronic Commerce*. ACM, 2006, pp. 1–7.

[2] Ali Alnoman. "Delay-aware Scheduling Scheme for Ubiquitous IoT Applications in Edge Computing". In: *2021 International Symposium on Networks, Computers and Communications (ISNCC)*. 2021, pp. 1–4. DOI: 10.1109/ISNCC52172.2021.9615690.

[3] Syed Umar Amin and M. Shamim Hossain. "Edge Intelligence and Internet of Things in Healthcare: A Survey". In: *IEEE Access* 9 (2021), pp. 45–59. DOI: 10.1109/ACCESS.2020.3045115.

[4] T. Bahreini, H. Badri, and D. Grosu. "An Envy-Free Auction Mechanism for Resource Allocation in Edge Computing Systems". In: *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. Oct. 2018, pp. 313–322.

[5] Alcardo Alex Barakabitze et al. "QoE Management of Multimedia Streaming Services in Future Networks: A Tutorial and Survey". In: *IEEE Communications Surveys Tutorials* 22.1 (2020), pp. 526–565. DOI: 10.1109/COMST.2019.2958784.

[6] Kashif Bilal and Aiman Erbad. "Edge computing for interactive media and video streaming". In: *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*. 2017, pp. 68–73. DOI: 10.1109/FMEC.2017.7946410.

[7] Ester Camina. "A generalized Assignment Game". In: *Mathematical Social Sciences, Elsevier* 52.2 (Sept. 2006), pp. 152–161.

[8] Matthew Cary et al. "Convergence of Position Auctions under Myopic Best-Response Dynamics". In: *ACM Transactions on Economics and Computation* 2.3 (July 2014), pp. 1–20.

[9] Matthew Cary et al. "Greedy Bidding Strategies for Keyword Auctions". In: *Proceedings of the 8th ACM Conference on Electronic Commerce*. EC '07. San Diego, California, USA: ACM, 2007, pp. 262–271. ISBN: 978-1-59593-653-0. DOI: 10.1145/1250910.1250949. URL: http://doi.acm.org/10.1145/1250910.1250949.

[10] Matthew Cary et al. "Greedy Bidding Strategies for Keyword Auctions". In: *Proceedings of the 8th ACM Conference on Electronic Commerce*. EC '07. San Diego, California, USA: ACM, 2007, pp. 262–271. ISBN: 978-1-59593-653-0. DOI: 10.1145/1250910.1250949. URL: http://doi.acm.org/10.1145/1250910.1250949.

[11] Xu Chen et al. "Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing". In: *IEEE/ACM Transactions on Networking* 24.5 (2016), pp. 2795–2808. DOI: 10.1109/TNET.2015.2487344.

[12] B. Edelman, M. Ostrovsky, and M. Schwarz. "Internet Advertising and the Generalized Second-Price Auction: Selling Billions of Dollars Worth of Keywords". In: *American Economic Review* 97.1 (2007), pp. 242–259.

[13] Benjamin Edelman and Michael Schwarz. "Optimal Auction Design and Equilibrium Selection in Sponsored Search Auctions". In: *American Economic Review* 100.2 (May 2010), pp. 597–602. DOI: 10.1257/aer.100.2.597. URL: https://www.aeaweb.org/articles?id=10.1257/aer.100.2.597.

[14] Guoju Gao et al. "Auction-based VM Allocation for Deadline-Sensitive Tasks in Distributed Edge Cloud". In: *IEEE Transactions on Services Computing* 14.6 (2021), pp. 1702–1716. DOI: 10.1109/TSC.2019.2902549.

[15] Hongzhi Guo and Jiajia Liu. "Collaborative Computation Offloading for Multiaccess Edge Computing Over Fiber–Wireless Networks". In: *IEEE Transactions on Vehicular Technology* 67.5 (2018), pp. 4514–4526. DOI: 10.1109/TVT.2018.2790421.

[16] Ummy Habiba, Setareh Maghsudi, and Ekram Hossain. "A Reverse Auction Model for Efficient Resource Allocation in Mobile Edge Computation Offloading". In: *2019 IEEE Global Communications Conference (GLOBECOM)*. 2019, pp. 1–6. DOI: 10.1109/GLOBECOM38437.2019.9014240.

[17] Hsiang-Jen Hong et al. "Optimizing Social Welfare for Task Offloading in Mobile Edge Computing". In: *2020 IFIP Networking Conference (Networking)*. 2020, pp. 524–528.

[18] Che-Wei Hsu, Yung-Lin Hsu, and Hung-Yu Wei. "Energy-Efficient Edge Offloading in Heterogeneous Industrial IoT Networks for Factory of Future". In: *IEEE Access* 8 (2020), pp. 183035–183050. DOI: 10.1109/ACCESS.2020.3029253.

[19] A. Jin, W. Song, and W. Zhuang. "Auction-Based Resource Allocation for Sharing Cloudlets in Mobile Cloud Computing". In: *IEEE Transactions on Emerging Topics in Computing* 6.1 (Jan. 2018), pp. 45–57.

[20] Hans Kellerer, Ulrich Pferschy, and David Pisinger. "Knapsack Problems". In: Springer, 2004. Chap. The Multiple-Choice Knapsack Problem.

[21] Terence Kelly. "Generalized knapsack solvers for multi-unit combinatorial auctions: analysis and application to computational resource allocation". In: *Proceedings of the 6th AAMAS international conference on Agent-Mediated Electronic Commerce: theories for and Engineering of Distributed Mechanisms and Systems*. July 2004, pp. 73–86.

[22] A. Kiani and N. Ansari. "Toward Hierarchical Mobile Edge Computing: An Auction-Based Profit Maximization Approach". In: *IEEE Internet of Things Journal* 4.6 (Dec. 2017), pp. 2082–2091.

[23] S. O. Krumkea and C.Thielen. "The generalized assignment problem with minimum quantities". In: *European Journal of Operational Research* 228.1 (2013), pp. 46–55.

[24] Śebastien Lahai and David M.Pennock. "Revenue analysis of a family of ranking rules for keyword auctions". In: *InProceedings of 8th ACM Conference on Electronic Commerce*. ACM, 2013, pp. 50–56.

[25] Tra Huong Thi Le et al. "Auction Mechanism for Dynamic Bandwidth Allocation in Multi-Tenant Edge Computing". In: *IEEE Transactions on Vehicular Technology* 69.12 (2020), pp. 15162–15176. DOI: 10.1109/TVT.2020.3036470.

[26] Haemin Lee et al. "Auction-based Deep Learning Computation Offloading for Truthful Edge Computing: A Myerson Auction Approach". In: *2021 International Conference on Information Networking (ICOIN)*. 2021, pp. 457–459. DOI: 10.1109/ICOIN50884.2021.9334016.

[27] Feixiang Li et al. "Auction Design for Edge Computation Offloading in SDN-based Ultra Dense Networks". In: *IEEE Transactions on Mobile Computing* 21.5 (2022), pp. 1580–1595. DOI: 10.1109/TMC.2020.3026319.

[28] Quanyi Li et al. "Reinforcement-Learning- and Belief-Learning-Based Double Auction Mechanism for Edge Computing Resource Allocation". In: *IEEE Internet of Things Journal* 7.7 (2020), pp. 5976–5985. DOI: 10.1109/JIOT.2019.2953108.

[29] Yuqing Li et al. "Online Cooperative Resource Allocation at the Edge: A Privacy-Preserving Approach". In: *2020 IEEE 28th International Conference on Network Protocols (ICNP)*. 2020, pp. 1–11. DOI: 10.1109/ICNP49622.2020.9259382.

[30] M. Liu and Y. Liu. "Price-Based Distributed Offloading for Mobile-Edge Computing With Computation Capacity Constraints". In: *IEEE Wireless Communications Letters* 7.3 (June 2018), pp. 420–423.

[31] P. Mach and Z. Becvar. "Mobile Edge Computing: A Survey on Architecture and Computation Offloading". In: *IEEE Communications Surveys Tutorials* 19.3 (2017), pp. 1628–1656.

[32] B. Al-Manthari et al. "Fair Class-Based Downlink Scheduling with Revenue Considerations in Next Generation Broadband Wireless Access Systems". In: *IEEE Transactions on Mobile Computing* 8.6 (June 2009), pp. 721–734.

[33] *Multi-access Edge Computing (MEC); MEC 5G Integration*. Group Report. Oct. 2020.

[34] *Multi-access Edge Computing (MEC); Study on MEC support for alternative virtualization*. Group Report. Nov. 2019.

[35] *Multi-access Edge Computing (MEC); Study on MEC support for alternative virtualization technologies*. Group Report. Nov. 2019.

[36] John Nash. "Non-Cooperative Games". In: *The Annals of Mathematics* 54.2 (Sept. 1951), pp. 286–295.

[37] *P.1411-11 - Propagation data and prediction methods for the planning of short-range outdoor radio communication systems and radio local area networks in the frequency range 300 MHz to 100 GHz*. Tech. Rep. Sept. 2021.

[38] Jianli Pan and James McElhannon. "Future Edge Cloud and Edge Computing for Internet of Things Applications". In: *IEEE Internet of Things Journal* 5.1 (2018), pp. 439–449. DOI: 10.1109/JIOT.2017.2767608.

[39] Quoc-Viet Pham et al. "Coalitional Games for Computation Offloading in NOMA-Enabled Multi-Access Edge Computing". In: *IEEE Transactions on Vehicular Technology* 69.2 (2020), pp. 1982–1993. DOI: 10.1109/TVT.2019.2956224.

[40] Tao Qin, Wei Chen, and Tie-Yan Liu. "Sponsored Search Auctions: Recent Advances and Future Directions". In: *ACM Trans. Intell. Syst. Technol.* 5.4 (Jan. 2015), 60:1–60:34. ISSN: 2157-6904.

[41] Ben Roberts et al. "Ranking and Tradeoffs in Sponsored Search Auctions". In: *In Proceedings of 14th ACM Conference on Electronic Commerce*. ACM, 2013, pp. 751–766.

[42] Alvin E. Roth and Marilda Sotomayor. "Chapter 16 Two-sided matching". In: vol. 1. Handbook of Game Theory with Economic Applications. Elsevier, 1992, pp. 485–541. DOI: https://doi.org/10.1016/S1574-0005(05)80019-0. URL: https://www.sciencedirect.com/science/article/pii/S1574000505800190.

[43] Michael H. Rothkopf. "Thirteen Reasons Why the Vickrey-Clarke-Groves Process Is Not Practical." In: *Operations Research* 55.2 (2007), pp. 191–197.

[44] Z. Sanaei et al. "Heterogeneity in Mobile Cloud Computing: Taxonomy and Open Challenges". In: *IEEE Communications Surveys Tutorials* 16.1 (Jan. 2014), pp. 369–392.

[45] M. Satyanarayanan. "The Emergence of Edge Computing". In: *Computer* 50.1 (Jan. 2017), pp. 30–39.

[46] Ziyu Shen, Jinshui Zhang, and Haisheng Tan. "A Truthful FPTAS Auction for the Edge-Cloud Pricing Problem". In: *2020 6th International Conference on Big Data Computing and Communications (BIGCOM)*. 2020, pp. 140–144. DOI: 10.1109/BigCom51056.2020.00027.

[47] W. Shi et al. "Edge Computing: Vision and Challenges". In: *IEEE Internet of Things Journal* 3.5 (Oct. 2016), pp. 637–646.

[48] Stavros Souravlas et al. "A Fair, Dynamic Load Balanced Task Distribution Strategy for Heterogeneous Cloud Platforms Based on Markov Process Modeling". In: *IEEE Access* 10 (2022), pp. 26149–26162. DOI: 10.1109/ACCESS.2022.3157435.

[49] Yi Su et al. "A Truthful Combinatorial Auction Mechanism towards Mobile Edge Computing in Industrial Internet of Things". In: *IEEE Transactions on Cloud Computing* (2022), pp. 1–1. DOI: 10.1109/TCC.2022.3155495.

[50] Sukhmani Sukhmani et al. “Edge Caching and Computing in 5G for Mobile AR/VR and Tactile Internet”. In: *IEEE MultiMedia* 26.1 (2019), pp. 21–30. DOI: 10.1109/MMUL.2018.2879591.

[51] W. Sun et al. “Double Auction-Based Resource Allocation for Mobile Edge Computing in Industrial Internet of Things”. In: *IEEE Transactions on Industrial Informatics* 14.10 (Oct. 2018), pp. 4692–4701.

[52] Tarik Taleb et al. “Mobile Edge Computing Potential in Making Cities Smarter”. In: *IEEE Communications Magazine* 55.3 (2017), pp. 38–43. DOI: 10.1109/MCOM.2017.1600249CM.

[53] David R. M. Thompson and Kevin Leyton-Brown. “Revenue optimization in the generalized second-price auction”. In: *InProceedings of 14th ACM Conference on Electronic Commerce*. ACM, 2013, pp. 837–852.

[54] Hal Varian. “Position auctions”. In: *International Journal of Industrial Organization* 25.6 (2007), pp. 1163–1178.

[55] William Vickrey. “Counterspeculation, auctions, and competitive sealed tenders”. In: *The Journal of Finance* 16.1 (Mar. 1961), pp. 8–37.

[56] *vSphere Resource Management*. Documentation. Mar. 2021. URL: https://docs.vmware.com/en/VMware-vSphere/7.0/vsphere-esxi-vcenter-server-702-resource-management-guide.pdf.

[57] Junjue Wang et al. “Bandwidth-Efficient Live Video Analytics for Drones Via Edge Computing”. In: *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. 2018, pp. 159–173. DOI: 10.1109/SEC.2018.00019.

[58] Quyuan Wang et al. “Incentive Mechanism for Edge Cloud Profit Maximization in Mobile Edge Computing”. In: *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*. 2019, pp. 1–6. DOI: 10.1109/ICC.2019.8761241.

[59] Quyuan Wang et al. “Profit Maximization Incentive Mechanism for Resource Providers in Mobile Edge Computing”. In: *IEEE Transactions on Services Computing* 15.1 (2022), pp. 138–149. DOI: 10.1109/TSC.2019.2924002.

[60] Rongfei Zeng et al. “FMore: An Incentive Scheme of Multi-dimensional Auction for Federated Learning in MEC”. In: *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. 2020, pp. 278–288. DOI: 10.1109/ICDCS47774.2020.00094.

[61] Heli Zhang et al. “Combinational Auction-Based Service Provider Selection in Mobile Edge Computing Networks”. In: *IEEE Access* 5 (2017), pp. 13455–13464. DOI: 10.1109/ACCESS.2017.2721957.

[62] Jing Zhang et al. “Joint Computation Offloading and Resource Allocation Optimization in Heterogeneous Networks With Mobile Edge Computing”. In: *IEEE Access* 6 (2018), pp. 19324–19337. DOI: 10.1109/ACCESS.2018.2819690.

[63] Jun Zhang and Khaled B. Letaief. “Mobile Edge Intelligence and Computing for the Internet of Vehicles”. In: *Proceedings of the IEEE* 108.2 (2020), pp. 246–261. DOI: 10.1109/JPROC.2019.2947490.

[64] Lei Zhang et al. “Joint Service Placement and Computation Offloading in Mobile Edge Computing: An Auction-based Approach”. In: *2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS)*. 2020, pp. 256–265. DOI: 10.1109/ICPADS51040.2020.00043.

[65] Chongyu Zhou and Chen-Khong Tham. “Where to Process: Deadline-aware Online Resource Auction in Mobile Edge Computing”. In: *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. 2018, pp. 675–680. DOI: 10.1109/PERCOMW.2018.8480192.

Appendix A: Proof of Propositions in Chapter 3

A.1 Proof of Proposition 1

Proof. As described before, for each task type n, the MEC server demands a price according to the bidding strategy given by (3.9). To prove the *individual rationality* property for the participating bidders, we need to show that the utility of each MEC server i is non-negative, i.e., $u_{i,n} \geq 0$.

Assume that some MEC (bidder) i wins some tasks from the task type n. Moreover, a total of $x_{i,n}$ units of resources are allocated at a unit price π_n. Therefore, the utility of MEC i for the allocation of a resource unit yields $u_{i,n} = (\pi_n - \theta_n v_i)\, x_{i,n}$. By definition, we have $x_{i,n} \geq 0$; Consequently, to establish the individual rationality, we need to show that $\pi_n \geq \theta_n v_i$. According to the payment rule given by (3.12), it is sufficient to show that

$$b_{i',n} \geq \theta_n v_i. \tag{A.1}$$

The MEC bidders are sorted in the decreasing order of their effective bids as

$$\eta_1 \geq \cdots \geq \eta_i \geq \eta_{i_0} \geq \eta_{i'} \geq \cdots \geq \eta_I.$$

Therefore, from the definition of effective bids in (3.10), we can con-

clude

$$b_{i,n} \leq b_{i_0,n} \leq b_{i',n}, \tag{A.2}$$

which implies $v_{i'} \geq v_{i_0} \geq v_i$.

According to Definition 5, the minimum bid that MEC server i' reports for task type n is given by

$$b_{i',n} \geq \theta_n v_{i'}. \tag{A.3}$$

Since $v_{i'} \geq v_i$ and $\theta \geq 0$, we can write

$$\theta_n v_{i'} \geq \theta_n v_i. \tag{A.4}$$

From (A.3) and (A.4), we obtain the individual rationality constraint as in (A.1). This ensures that no bidder incurs a loss from participating in the proposed auction mechanism. ■

A.2 Proof of Proposition 2

Proof. To prove that the proposed auction mechanism is *locally envy-free*, we need to show that no MEC server can profitably be re-matched with the position (task type) just right above to the one she is currently assigned to. In our system model, the auction positions represent the different types of offloading requests, which are determined based on the service providers' offered application types. In the proposed auction mechanism, the resource allocation prices π_n for the offloading tasks of type n are identical irrespective of the MEC servers to whom they are assigned. Therefore, we need to verify the locally envy-free property for the auction scenario where $N \geq 1$. Moreover, the number of auction positions is limited by the number of MEC servers, i.e., $I \geq N$.

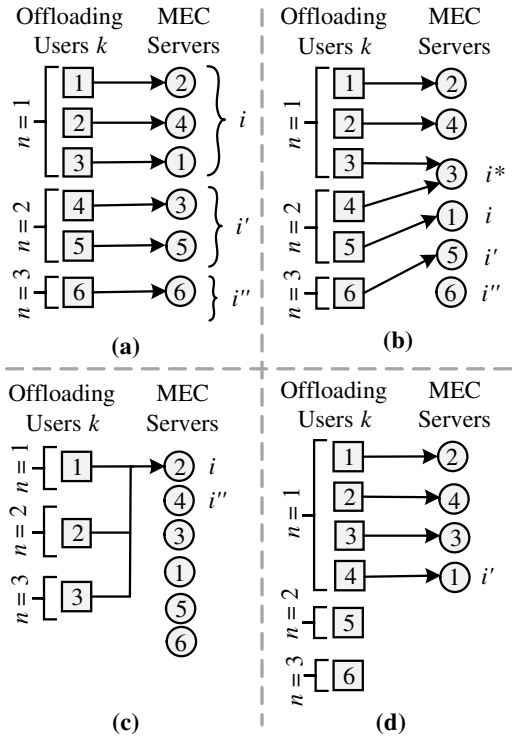


Figure A.1: Feasible outcomes to verify envy-free allocation

Hence, we consider $N = 3$ to investigate the locally envy-free property for the proposed auction mechanism. In Fig. A.1, we illustrate different *feasible outcomes*[1], which can be classified into the following three scenarios: (i) The number of MEC servers is exactly same as the number of offloading tasks, i.e., $I = K$ (e.g., Fig. A.1(a) and (b)); (ii) There are more MEC servers than the requested offloading tasks, i.e., $I > K$ (e.g., Fig. A.1(c)); (iii) There is strong competition among the bidders (MEC servers) as $I < K$ (e.g., Fig. A.1(d)).

Case 1: We first show the locally envy-freeness for the auction scenario when there are equal number of MEC servers and offloading tasks. Moreover, there is one-to-one matching for the task assignment as de-

[1]A feasible outcome μ represents the outcome of matching an object to a seller, which is compatible with the allocation outcome pair $(\boldsymbol{x}, \boldsymbol{\pi})$. At a feasible assignment, each seller (MEC server) obtains some utility $u_{i,n} \geq 0$ for allocating $x_{i,n} \geq 0$ units at price $\pi_n \geq 0$ for each object (offloading tasks). If the seller gets no allocation, her utility is zero. In other words, a feasible outcome is individually rational [7].

picted in Fig. A.1(a). Let MEC servers i, i', and i'' win the task types $n = 1$, $n = 2$, and $n = 2$, respectively. Then we need to verify the following two cases

- *Case 1-A*: Let us exchange the allocation of bidder i' with bidder i; i.e., bidder i' is re-matched with allocation of $x_{i',1}$ units at price π_1. Thus, to establish the locally envy-free property, we need to ensure that the utility of MEC server i' does not improve with this exchange; formally,

$$(\pi_2 - \theta_2 v_{i'})\, x_{i',2} \geq (\pi_1 - \theta_1 v_{i'})\, x_{i',1}. \tag{A.5}$$

  According to the pricing rule in (3.12), we get

$$\pi_1 = b_{i',1}, \text{ and } \pi_2 = b_{i'',2}, \text{ and } \pi_3 = b_{i'',3} + \epsilon, \tag{A.6}$$

  where ϵ is a very small positive constant. By substituting the prices given by (A.6) into (A.5), we obtain the following condition that is sufficient to show the locally envy-freeness:

$$b_{i',1} x_{i',1} - b_{i'',2} x_{i',2} \leq (\theta_1 x_{i',1} - \theta_2 x_{i',2})\, v_{i'}. \tag{A.7}$$

  According to the bidding strategy in (3.9), we have

$$b_{i',1} = \theta_1 v_{i'}, \text{ and } b_{i',2} \geq \theta_2 v_{i'}.$$

  Therefore,

$$b_{i',1}\, x_{i',1} - b_{i'',2} x_{i',2} \leq \theta_1 v_{i'} x_{i',1} - \theta_2 v_{i''} x_{i',2}. \tag{A.8}$$

  Moreover, we know that $v_{i''} \geq v_{i'}$. Hence,

$$\theta_1 v_{i'} x_{i',1} - \theta_2 v_{i''} x_{i',2} \leq \theta_1 v_i x_{i',1} - \theta_2 v_{i'} x_{i',2}. \tag{A.9}$$

Therefore, the condition in (A.7) is satisfied and the locally envy-free property holds for this case.

- *Case 1-B*: Let the allocation of bidder i'' is exchanged with that of bidder i', when $x_{i'',2}$ units are allocated to bidder i'' at price π_2. We need to show that the utility of MEC server i'' does not improve due to this exchange, i.e.,

$$(\pi_3 - \theta_3 v_{i''})\, x_{i'',3} \geq (\pi_2 - \theta_2 v_{i''})\, x_{i'',2}. \tag{A.10}$$

By substituting the payment prices as given by (A.5), we obtain the condition for locally envy-freeness as following

$$b_{i'',2}\, x_{i'',2} - (b_{i'',3} + \epsilon) x_{i'',3} \leq (\theta_2 x_{i'',2} - \theta_3 x_{i'',3}) v_{i''}. \tag{A.11}$$

Due to the assumption of the exchange of allocation, we have $x_{i'',1}^{(t-1)} = x_{i'',2}^{(t-1)} = 0$ for bidder i''. Thus, the bidding strategy in (3.9) yields

$$b_{i'',2} = \theta_2 v_{i''} + \frac{x_{i'',1}^{(t-1)} + x_{i'',2}^{(t-1)}}{2}(\pi_1^{(t-1)} - \theta_1 v_{i''}) = \theta_2 v_{i''} \tag{A.12}$$

By a similar argumentation, we conclude

$$b_{i'',3} = \theta_3 v_{i''} + B^*, \tag{A.13}$$

where $B^* = \frac{x_{i'',2}^{(t-1)} + x_{i'',3}^{(t-1)}}{2}(\pi_2^{(t-1)} - \theta_2 v_{i''}) > 0$.

Using (A.12) and (A.13), we can write

$$\begin{aligned} b_{i'',2}\, x_{i'',2} - b_{i'',3}\, x_{i'',3} = \theta_2 v_{i''} x_{i'',2} \\ - (\theta_3 v_{i''} + B^* + \epsilon)\, x_{i'',3} \end{aligned} \tag{A.14}$$

Hence it becomes evident that

$$\theta_2 v_{i''} x_{i'',2} - (\theta_3 v_{i''} + B^* + \epsilon)\, x_{i'',3} \leq (\theta_2 x_{i'',2} - \theta_3 x_{i'',3}) v_{i''} \tag{A.15}$$

From (A.14) and (A.15), we can conclude that the locally envy-free condition in (A.11) holds for this case.

Case 2: Here, we investigate the many-to-one matching scenario when there are equal number of MEC servers and offloading tasks. We consider MEC server i^* as the bidder who wins tasks of multiple task types, as shown in Fig. A.1(b). Here, we consider MEC server i wins the task of type $n = 2$ and MEC server i' wins the task of type $n = 3$. Moreover, we assume that MEC server i'' loses the auction and receives no allocation.

The exchange of allocation between the bidders i and i' cannot improve their utilities, which can be shown by following the similar steps as in Case 1-B. Therefore, we verify the following two cases:

- *Case 2-A*: At first, we show that bidder i cannot improve her utility by exchanging her allocation with bidder i^* who is ranked above her; i.e.,

$$(\pi_2 - \theta_2 v_i)\, x_{i,2} \geq (\pi_1 - \theta_1 v_i)\, x_{i,1} + (\pi_2 - \theta_2 v_i)\, x_{i,2}, \tag{A.16}$$

which, by $x_{i,1} \geq 0$, is equivalent to

$$\pi_1 - \theta_1 v_i \leq 0 \tag{A.17}$$

By the pricing rule in (3.12), we have

$$\pi_1 = b_{i',1}, \text{ and } \pi_2 = b_{i'',2}, \text{ and } \pi_3 = b_{i'',3}. \tag{A.18}$$

Moreover, from the bidding strategy given by (3.9), we can conclude that

$$b_{i',1} = \theta_1 v_{i'},\ b_{i',2} \geq \theta_2 v_{i'},\ \text{and}\ b_{i'',3} \geq \theta_3 v_{i''}. \tag{A.19}$$

From (A.18) and (A.19), it is obvious that $\pi_1 = b_{i',1} = \theta_1 v_{i'}$ and thus locally envy-free condition in (A.17) holds for this case.

- *Case 2-B*: Next, we show that bidder i'' cannot improve her utility by exchanging her allocation with bidder i' ranked above her, i.e.,

$$(\pi_3 - \theta_3 v_{i''})\, x_{i'',3} \leq 0. \tag{A.20}$$

Alternatively, we can write

$$\pi_3 \leq \theta_3\, v_{i''}. \tag{A.21}$$

From the fact that $x^{(t-1)}_{i'',2} = x^{(t-1)}_{i'',3} = 0$ in this scenario, we conclude from the bidding strategy (3.9) that $\pi_3 = b_{i'',3} = \theta_3 v_{i''}$. Therefore, the the sufficient condition for locally envy-freeness in (A.21) is satisfied.

Case 3: When there is more MEC servers than the offloading tasks, the bidders compete to win the tasks from a higher level of task types. Consequently, every bidder i tends to bid lower as minimum as $\theta_n v_i$. Let us consider such a competitive scenario where the first-ranked MEC server (e.g., bidder i) wins all the tasks of type $n = 1$, $n = 2$, and $n = 3$. Fig. A.1 (c) illustrates such scenario. We consider the MEC server i'' as the bidder ranked next to bidder i and loses the auction with no allocation.

In order to establish the locally envy-free property, we need to show

that bidder i'' cannot improve her utility by exchanging its allocation with bidder i. Formally,

$$\begin{aligned}(\pi_1 - \theta_1 v_{i''})\, x_{i'',1} + (\pi_2 - \theta_2 v_{i''})\, x_{i'',2} \\ + (\pi_3 - \theta_3 v_{i''})\, x_{i'',3} \ \leq 0,\end{aligned} \tag{A.22}$$

or equivalently,

$$\begin{aligned}\pi_1 x_{i'',1} + \pi_2 x_{i'',2} + \pi_3 x_{i'',3} \leq \\ (\theta_1 x_{i'',1} + \theta_2 x_{i'',2} + \theta_3 x_{i'',3}) v_{i''}\end{aligned} \tag{A.23}$$

According to the pricing rule in (3.12), we have

$$\pi_1 = b_{i'',1}, \pi_2 = b_{i'',2}, \text{ and } \pi_3 = b_{i'',3}. \tag{A.24}$$

Since bidder i'' didn't win any task before the assumption of exchange of allocation, we have $x^{(t-1)}_{i'',1}, x^{(t-1)}_{i'',2} = x^{(t-1)}_{i'',3} = 0$. By following the bidding strategy in (3.9), we obtain

$$b_{i'',1} = \theta_1 v_{i''}, b_{i'',2} = \theta_2 v_{i''}, \text{ and } b_{i'',3} = \theta_3 v_{i''}. \tag{A.25}$$

Therefore we have

$$\begin{aligned}\pi_1 x_{i'',1} + \pi_2 x_{i'',2} + \pi_3 x_{i'',3} = \\ (\theta_1 x_{i'',1} + \theta_2 x_{i'',2} + \theta_3 x_{i'',3}) v_{i''},\end{aligned} \tag{A.26}$$

which satisfies the locally envy-free condition in (A.23).

Case 4: Finally, we show that locally envy-free property holds also for the case where the number of incoming requests for offloading tasks is larger than the available MEC servers. In such a scenario, there can be some unassigned offloading tasks when all of the MEC servers are assigned for computing the offloading tasks with higher resource demand and higher position in task popularity (i.e., task type n). The

unassigned tasks don't affect the utilities of the winning bidders. Here, all of the bidders win some tasks. The proof for locally envy-freeness is similar to the cases 1 and 2-A if the MEC servers are assigned to tasks of different task type.

In the extreme case of the aforementioned scenario, all the resources of MEC servers are allocated to the tasks of type $n = 1$. This satisfies the users' excessive demand for computational resources and all other tasks remain unassigned, as depicted in Fig. A.1 (d). Also in this case, the locally envy-free property still holds. This is due to the following reason: Since only the tasks of type $n = 1$ are allocated, all the winning bidders receive the same payment prices $\pi_1 = b_{i',1} + \epsilon$, where i' is the last bidder who wins the auction. Thus no bidder can improve her utilities by exchanging allocations. Consequently, we can conclude that the proposed auction mechanism is locally envy-free. ■

A.3 Proof of Proposition 3

Proof. In **Algorithm 1**, the sorting of MEC servers can be performed in $O(I \log I)$ time (e.g. quicksort algorithm). Similarly, the sorting of user list takes $O(K \log K)$ time for each task type n. When the algorithm calls Greedy allocation procedure for each type n, **Algorithm 2** takes $\mathcal{O}(K)$ time. As **Algorithm 1** calls the Greedy allocation procedure N times, in the worst-case, $O(I \log I) + O(NK + NK \log K)$ operations are needed. Thus, the proposed approximate solution for $\mathcal{P}_1$ can be obtained in polynomial time. ■

Appendix B: Proof of Theorems in Chapter 4

B.1 Proof of Theorem 1

Proof. To prove the individual rationality property for the proposed MEC offloading auction mechanism, we need to show that every server achieves a non-negative utility gain for each of its VM, i.e., $u^n_{i,r_m}(t) \geq 0$, for each processor n in any time slot t.

According to (4.9), the utility of the VM allocated to position $m = s$ at server i is given by:

$$u^n_{i,r_s}(t) = \sum_{s=1}^{K} \lambda^n_{s_j} \theta^n_{i,r_s}(t) \left(p^n_s(t) - v^n_{i,r_s}\right) x^n_{s,r_{is}}(t). \tag{B.1}$$

By definition, we know that $\lambda^n_{s_j} \geq 0$, $\theta^n_{i,r_s}(t) \geq 0$, and $x^n_{s,r_{is}}(t) \geq 0$. Hence, the sufficient condition to satisfy the individual rationality property yields,

$$p^n_s(t) - v^n_{i,r_s} \geq 0. \tag{B.2}$$

The GSP pricing rule as proposed in (4.21) guarantees that $p^n_s(t) \geq b^n_{i,r_s}(t)$. Moreover, the adaptive RBB strategy, ensures that no server bids lower than actual valuation of a VM, by restricting the server to choose the target position in a way that the VM does not deviate from the current ranking position. This results into a positive value for

the bid adjustment factor $\left(p^n_{s^*_m}(t-1) - v^n_{i,r_{(m=s)}}\right)$ in (4.22), and consequently satisfies $b^n_{i,r_s}(t) \geq v^n_{i,r_s}$. Therefore, using the transitive property of inequalities, we can conclude $p^n_s(t) \geq v^n_{i,r_s}$, satisfying the individual rationality condition in (B.2). ■

B.2 Proof of Theorem 2

Proof. To prove the existence of symmetric Nash equilibrium (SNE), we need to show that the set of allocation prices $p^n_s(t)$ satisfies the SNE inequalities in (4.23) for all task positions $s \in \mathcal{K}^n(t)$ in any given time slot t. This means, every VM allocated to some task position s at processor n in the time slot t, is better off with the resulting utility gain and does not wish to exchange its current allocation with any other position within the processor.

As shown in [54], when GSP allocation prices satisfy the SNE condition for the positions $s+1$ and $s-1$, then it satisfies the inequalities for all positions s. Therefore, we verify SNE conditions considering two cases: (1) exchanging task allocation between positions $(s-1)$ and s; and then (2) exchanging between positions s and $(s+1)$.

***Case 1*:** Let task positions $s-1$ and s be originally assigned to VMs at the ranks $m = s-1$ and $m = s$, respectively (i.e., $r^n_{i(s-1)}$ and r^n_{is}), at a given time slot t. Thus, after the exchange, VM $r^n_{i(s-1)}$ will be matched to the s-th position with priority score $\lambda^n_{s_j}$ and the allocation price $p^n_s(t)$; and VM r^n_{is} will be matched to the $(s-1)$-th task position with priority score $\lambda^n_{s-1_j}$ and a allocation price of $p^n_{s-1}(t)$.

The SNE condition for the positions $s-1$ and s requires the following inequalities to be satisfied:

$$\lambda^n_{(s-1)_j}\theta^n_{i,r_{s-1}}(t)\left(p^n_{s-1}(t) - v^n_{i,r_{s-1}}\right) \geq \lambda^n_{s_j}\theta^n_{i,r_{s-1}}(t) \left(p^n_s(t) - v^n_{i,r_{s-1}}\right), \tag{B.3}$$

$$\lambda^n_{s_j}\theta^n_{i,r_s}(t)\left(p^n_s(t) - v^n_{i,r_s}\right) \geq \lambda^n_{(s-1)_j}\theta^n_{i,r_s}(t)\left(p^n_{s-1}(t) - v^n_{i,r_s}\right) \tag{B.4}$$

Now, we can re-write the SNE condition by combining inequalities (B.3) and (B.4) as:

$$\left(\lambda^n_{(s-1)_j} - \lambda^n_{s_j}\right) v^n_{i,r_s} \geq \lambda^n_{(s-1)_j}p^n_{s-1}(t) - \lambda^n_{s_j}p^n_s(t) \geq \left(\lambda^n_{(s-1)_j} - \lambda^n_{s_j}\right) v^n_{i,r_{s-1}} \tag{B.5}$$

Besides, by definition $\lambda^n_{(s-1)_j} \geq \lambda^n_{s_j} \geq \lambda^n_{(s+1)_j} \geq 0$, and by GSP pricing rule we have $p^n_{s-1}(t) \geq p^n_s(t) \geq p^n_{s+1}(t)$. Therefore, we obtain $\lambda^n_{s_j} p^n_{s-1}(t) \geq \lambda^n_{s_j} p^n_s(t)$, which results,

$$\lambda^n_{(s-1)_j} p^n_{s-1}(t) - \lambda^n_{s_j} p^n_s(t) \geq \left(\lambda^n_{(s-1)_j} - \lambda^n_{s_j}\right) p^n_{s-1}(t). \tag{B.6}$$

According to GSP pricing rule (4.21), the allocation prices for the task positions $(s-1)$ and s result as

$$\begin{aligned} p^n_{s-1}(t) &= \Theta^n_{R_{(s-1)}} b^n_{i,r_s}(t) \geq b^n_{i,r_{s-1}}(t), \text{ and} \\ p^n_s(t) &= \Theta^n_{R_s} b^n_{i,r_{s+1}}(t) \geq b^n_{i,r_s}(t), \end{aligned} \tag{B.7}$$

respectively. Moreover, the proposed RBB strategy (4.22) ensures that $b^n_{i,r_{s-1}} \geq v^n_{i,r_{s-1}}$. So, we can obtain $p^n_{s-1}(t) \geq v^n_{i,r_{s-1}}$, and rewrite (B.6) as

$$\lambda^n_{(s-1)_j} p^n_{s-1}(t) - \lambda^n_{s_j} p^n_s(t) \geq \left(\lambda^n_{(s-1)_j} - \lambda^n_{s_j}\right) v^n_{i,r_{s-1}}, \tag{B.8}$$

satisfying the right side of the SNE inequality in (B.5).

Next, using GSP-based prices in (B.7) we obtain,

$$\begin{aligned} &\lambda^n_{(s-1)_j} p^n_{s-1}(t) - \lambda^n_{s_j} p^n_s(t) \\ &= \lambda^n_{(s-1)_j} \Theta^n_{R_{(s-1)}} b^n_{i,r_s}(t) - \lambda^n_{s_j} \Theta^n_{R_s} b^n_{i,r_{s+1}}(t), \end{aligned}$$

where, price adjustment rates satisfies: $\Theta^n_{R_{(s-1)}} \geq \Theta^n_{R_s} \geq \Theta^n_{R_{(s+1)}} \geq 1$, according to our modified GSP pricing mechanism. Besides, the ranking order of VMs ensures that

$$b^n_{i,r_1}(t) \ldots \leq b^n_{i,r_{s-1}}(t) \leq b^n_{i,r_s}(t) \leq b^n_{i,r_{s+1}}(t) \leq \ldots b^n_{i,r_K}(t).$$

Thus, we can apply $\left(\Theta^n_{R_{(s-1)}} b^n_{i,r_s}(t) \leq \Theta^n_{R_s} b^n_{i,r_{(s+1)}}(t)\right)$ into (B.8), which yields:

$$\begin{aligned} &\lambda^n_{(s-1)_j} \Theta^n_{R_{(s-1)}} b^n_{i,r_s}(t) - \lambda^n_{s_j} \Theta^n_{R_s} b^n_{i,r_{s+1}}(t) \\ &\qquad \leq \left(\lambda^n_{(s-1)_j} - \lambda^n_{s_j}\right) \Theta^n_{R_{(s-1)}} b^n_{i,r_s}(t) \end{aligned} \tag{B.9}$$

Moreover, the proposed RBB strategy ensures $b^n_{i,r_s}(t) \geq v^n_{i,r_s}$. So, even if the server bids the minimum and and gets the same as the bid (i.e., $(\Theta_{R_{(s-1)}} = 1)$, the SNE condition in the left side of the inequality (B.5) is still satisfied. This means,

$$\lambda^n_{(s-1)_j}\Theta^n_{R_{(s-1)}}b^n_{i,r_s}(t) - \lambda^n_{s_j}\Theta^n_{R_s}b^n_{i,r_{s+1}}(t) \leq (\lambda^n_{(s-1)_j} - \lambda^n_{s_j})v^n_{i,r_s} \tag{B.10}$$

***Case 2*:** Now, we consider the exchange of allocations between the task positions s and $s+1$, and verify the SNE condition. In this case, the VM $r^n_{i(s)}$ will be matched to the $s+1$-th position with priority score $\lambda^n_{s+1_j}$ and the allocation price $p^n_{s+1}(t)$; and VM $r^n_{i(s+1)}$ will be matched to the (s)-th task position with priority score $\lambda^n_{s_j}$ and a allocation price of $p^n_s(t)$. The SNE condition for this case thus becomes:

$$\left(\lambda^n_{s_j} - \lambda^n_{(s+1)_j}\right) v^n_{i,r_{s+1}} \geq \lambda^n_{s_j}p^n_s(t) - \lambda^n_{(s+1)_j}p^n_{s+1}(t) \geq \left(\lambda^n_{s_j} - \lambda^n_{(s+1)_j}\right) v^n_{i,r_s} \tag{B.11}$$

The GSP pricing rule (4.21) results the allocation prices for positions s and $s+1$ as,

$$\begin{aligned} p^n_s(t) &= \Theta^n_{R_s}b^n_{i,r_{s+1}}(t) \geq b^n_{i,r_s}(t), \text{ and} \\ p^n_{s+1}(t) &= \Theta^n_{R_{(s+1)}}b^n_{i,r_{s+2}}(t) \geq b^n_{i,r_{s+1}}(t) \end{aligned} \tag{B.12}$$

Similar to Case 1, we obtain $p^n_s(t) \geq v^n_{i,r_s}$ by the proposed RBB strategy which ensures $b^n_{i,r_s}(t) \geq v^n_{i,r_s}$. Thus, we get

$$\lambda^n_{s_j}p^n_s(t) - \lambda^n_{(s+1)_j}p^n_{s+1}(t) \geq \left(\lambda^n_{s_j} - \lambda^n_{(s+1)_j}\right) v^n_{i,r_s},$$

satisfying the right side of the SNE inequality (B.11).

Besides, the ranking order by the proposed GSP mechanism, which

still holds for this case, gives us $\left(\Theta^n_{R_s} b^n_{i,r_{(s+1)}}(t) \leq \Theta^n_{R_{(s+1)}} b^n_{i,r_{(s+2)}}(t)\right)$. Thus, we can write:

$$\begin{aligned}\lambda^n_{s_j} \Theta^n_{R_s} b^n_{i,r_{(s+1)}}(t) - \lambda^n_{(s+1)_j} \Theta^n_{R_{(s+1)}} b^n_{i,r_{s+2}}(t) \\ \leq \left(\lambda^n_{s_j} - \lambda^n_{(s+1)_j}\right) \Theta^n_{R_s} b^n_{i,r_{(s+1)}}(t)\end{aligned} \tag{B.13}$$

This completes the SNE condition in (B.11) for this case, as the proposed RBB strategy ensures $b^n_{i,r_{(s+1)}}(t) \geq v^n_{i,r_{(s+1)}}$ and satisfies the following, even if the server gets the minimum payment (i.e., $\Theta^n_{R_{(s)}} = 1$),

$$\begin{aligned}\lambda^n_{s_j} \Theta^n_{R_s} b^n_{i,r_{(s+1)}}(t) - \lambda^n_{(s+1)_j} \Theta^n_{R_{(s+1)}} b^n_{i,r_{s+2}}(t) \\ \leq \left(\lambda^n_{s_j} - \lambda^n_{(s+1)_j}\right) v^n_{i,r_{(s+1)}}.\end{aligned} \tag{B.14}$$

Finally, we can conclude that the proposed GSP allocation prices satisfies the SNE condition for all task positions. Subsequently, the mechanism reaches to the SNE point at each processor n, when every bidder (i.e. server) follows the proposed RBB strategy in any time slot t. ■

B.3 Proof of Theorem 3

Proof. Let, VMs r_A, r_B, and r_C are allocated to the tasks in positions $s-1$, s, and $s+1$, respectively in some time slot t. The ranking order by the proposed GSP mechanism ensures,

$$\begin{aligned}&\Theta^n_{R_{(s-1)}} \geq \Theta^n_{R_s} \geq \Theta^n_{R_{(s+1)}} \geq 1, \text{ and} \\ &b^n_{i,r_A} \leq b^n_{i,r_B} \leq b^n_{i,r_C}.\end{aligned} \tag{B.15}$$

Besides, the allocation prices for the positions $(s-1)$ and s, which satisfy the SNE condition during the time slot t are given by,

$$p^n_{(s-1)} = \Theta^n_{R_{(s-1)}} b^n_{i,r_B}, \text{ and} \tag{B.16}$$

$$p^n_s = \Theta^n_{R_s} b^n_{i,r_C}. \tag{B.17}$$

After reaching the SNE at time slot t, a server can still adjust the bid for the next round, while maintaining the current payment or rank position. So, the proposed GSP mechanism would result into a new set of auction outcome reaching to another equilibrium point in SNE. We can derive the upper bound to the SNE, by considering the scenario when a server i sets the bid for VM r_B in a way so it can move one slot up by beating VM r_A and make at least as much profit as the VM r_B makes now. The highest break-even bid for this case satisfies the following condition:

$$\begin{aligned} &\lambda^n_{(s-1)_j} \left(p^{'n}_{(s-1)} - v^n_{i,r_B} \right) x^n_{(s-1),r_A} \\ &\qquad = \lambda^n_{s_j} \left(p^n_s - v^n_{i,r_B} \right) x_{s,r_B} \end{aligned} \tag{B.18}$$

where $p^{'n}_{(s-1)}$ is the amount that the VM r_B believes to get after moving up, and p^n_s is the amount that VM currently gets for the task position s. We can re-write eqn. (B.18) as,

$$\begin{aligned} &\lambda^n_{(s-1)_j} \left(\Theta^n_{R_{(s-1)}} b^{'n}_{i,r_B} - v^n_{i,r_B} \right) x^n_{(s-1),r_A} \\ &\qquad = \lambda^n_{s_j} \left(\Theta^n_{R_s} b^n_{i,r_C} - v^n_{i,r_B} \right) x^n_{s,r_B}. \end{aligned} \tag{B.19}$$

Solving for b'^{n}_{i,r_B} gives us the highest bid ensuring that VM r_B gets the profit as minimum as the current one.

$$b'^{n}_{i,r_B} x^{n}_{(s-1),r_A} = \frac{v^{n}_{i,r_B}}{\Theta^{n}_{R_{(s-1)}}} x^{n}_{(s-1),r_A} + \frac{\lambda^{*}\Theta^{n}_{R_s}}{\Theta^{n}_{R_{(s-1)}}} \left(b^{n}_{i,r_C} - v^{n}_{i,r_B}\right) x^{n}_{s,r_B} \tag{B.20}$$

where $\lambda^{*} = \lambda^{n}_{s_j}/\lambda^{n}_{(s-1)_j}$. So, the general expression of the upper bound to the SNE can be written as,

$$b^{n,\mathrm{UB}}_{i,r_s} x^{n}_{(s-1),r_{(s-1)}} = \frac{v^{n}_{i,r_s}}{\Theta^{n}_{R_{(s-1)}}} x^{n}_{(s-1),r_{(s-1)}} + \frac{\lambda^{*}\Theta^{n}_{R_s}}{\Theta^{n}_{R_{(s-1)}}} \left(b^{n}_{i,r_{(s+1)}} - v^{n}_{i,r_s}\right) x^{n}_{s,r_s} \tag{B.21}$$

Next, in order to find the lower bound to the SNE we consider the scenario when a server i bids defensively, with the belief that if he/she bids too low then it will squeeze the profit of the VM above so much that the opponent might prefer moving down to lower position. The lowest break-even bid for this case, satisfies the condition to match the profit that VM above makes right now and the profit that VM in position $(s-1)$ would make if it goes down to the position s, i.e.:

$$\lambda^{n}_{(s-1)_j} \left(p^{*n}_{(s-1)} - v^{n}_{i,r_A}\right) x^{n}_{(s-1),r_A} = \lambda^{n}_{s_j} \left(p'^{n}_{s} - v^{n}_{i,r_A}\right) x^{n}_{s,r_B} \tag{B.22}$$

where $p^{*n}_{(s-1)}$ is the amount that the VM r_A currently gets from being in the position $(s-1)$, and p'^{n}_{s} represents the amount that the VM r_A would get if it moves down to the position s. Thus, (B.22) becomes,

$$\lambda^{n}_{(s-1)_j} \left(\Theta^{n}_{R_{(s-1)}} b^{*n}_{i,r_B} - v^{n}_{i,r_A}\right) x^{n}_{(s-1),r_A} = \lambda^{n}_{s_j} \left(\Theta^{n}_{R_s} b^{n}_{i,r_C} - v^{n}_{i,r_A}\right) x^{n}_{s,r_B} \tag{B.23}$$

Solving for b^{*}_{i,r_B} leads to the lower bound to the SNE,

$$b^{*n}_{i,r_B} x^{n}_{(s-1),r_A} = \frac{v^{n}_{i,r_A}}{\Theta^{n}_{R_{(s-1)}}} x^{n}_{(s-1),r_A} + \frac{\lambda^{*} \Theta^{n}_{R_s}}{\Theta^{n}_{R_{(s-1)}}} \left(b^{n}_{i,r_C} - v^{n}_{i,r_A} \right) x_{s,r_B} \tag{B.24}$$

Therefore, the general expression of the lower bound of SNE yields:

$$b^{n,\mathrm{LB}}_{i,r_s} x^{n}_{(s-1),r_{(s-1)}} = \frac{v^{n}_{i,r_{(s-1)}}}{\Theta^{n}_{R_{(s-1)}}} x^{n}_{(s-1),r_{(s-1)}} + \frac{\lambda^{*} \Theta^{n}_{R_s}}{\Theta^{n}_{R_{(s-1)}}} \left(b^{n}_{i,r_{(s+1)}} - v^{n}_{i,r_{(s-1)}} \right) x_{s,r_s}. \tag{B.25}$$

■

B.4 Proof of Theorem 4

Proof. The **Algorithm 4** is run at N processors simultaneously, which determines N distinct sets of resource allocation outcomes for different processors in every time slot t. In the first step, the incoming offloading requests at the processor n are gathered into respective task positions which takes a constant time in the order of $O(1)$. After initiating the request for bids to I servers, the algorithm updates the computation performance quality scores of the available VMs where the loop function runs $(I \times R^n)$ times. Upon receiving the bids from servers, the algorithm obtains the resource allocation and pricing decisions using the **Algorithm 3**.

The **Algorithm 3** takes $O(K \log K)$ and $O(R^n \log R^n)$ time to sort the offloading tasks, VMs and price adjustment rates, respectively. Then, corresponding task assignment and pricing decisions are then made in a constant time in the order of $O(1)$. So, the overall time complexity of **Algorithm 3** becomes $O(R^n \log R^n)$ for each processor.

After **Algorithm 3** returns the results back, the **Algorithm 4** finishes the remaining task of updating the workloads of VMs in $O(1)$ time. Finally, the worst-case time complexity of the **Algorithm 4** becomes $O(IR^n)$, ignoring the lower order of $O(R^n \log R^n)$.

On the other side, each server uses the **Algorithm 5** to determine his/her bids for each processor during time slot t. In this algorithm, each server requires $O(R_i^n)$ time to find the target slot, and then a constant time of order $O(1)$ to update the bid for each VM. So, the **Algorithm 5** also has a polynomial time complexity of $O(R_i^n)$ for each processor n.

During the time slot t, when the overall offloading service provisioning process is performed, the algorithms are run at N processors, the same number of operations are repeated across N processors and I servers. So, we can conclude that the proposed GSP-based MEC offloading mechanism is computationally efficient as the polynomial time complexity holds for every time slot. ■