mISO: Incentivizing Demand-Agnostic Microservices for Edge-Enabled IoT Networks

Amit Samanta, Quoc-Viet Pham, Nhu-Ngoc Dao, Ammar Muthanna, and Sungrae Cho

Abstract—The recent expansion of mobile IoT devices (MIoTDs) along with the exposure of many compute-intensive and latency-critical applications, have given a step rise to the mobile edge computing (MEC) platform to process computational microservices at the edge. The paramount importance of designing an effective incentive mechanism is a very important topic for such systems to get a fair amount of resources and provide incentives to MIoDs. Hence, we design a MEC platform with heterogeneous MIoTDs participating in a computational microservice offloading scheme. Here, we propose an incentive approach applying a double auction mechanism to incentivize the involvement of MIoTDs. In practice, the incentive mechanism typically interacts with the demand estimation scheme that estimates the demand profile of MIoTDs. As a result, we design a novel mechanism for microservices – *microservice Incentive Service Offloading (mISO)*, which comprises an incentive approach and a demand estimation scheme. The mISO mechanism holds truthfulness, rationality, and low computational complexity while guaranteeing positive social welfare and generating the optimal demand profiles for MIoTDs. Simulation results showed that mISO provides 18–21 % and 25–30 % improvements in terms of average latency and resource utilization compared to existing works.

Index Terms—Auction game, edge computing, demand-agnostic, microservice offloading, incentive mechanism.

1 Introduction

Mobile Edge Computing (MEC) [1]–[4] has been considered as an efficient and important paradigm for compute-intensive and latency-stringent Internet of Things (IoT) applications [5]. Such kind of infrastructure incorporates several advantages to edge devices, while offering distributed decision-making and privacy-preserving mechanisms for mobile IoT devices (MIoTDs). Here, the tasks (i.e., processing and data-dependent tasks) and microservices are transferred to the network edge [6], [7] in order to access the edge servers by the MIoTDs [8], [9] through wireless connections. The mobile devices offload their microservices at the network edge, while maintaining low microservice latency. Since the edge servers are deployed on a small scale at the network edge, the resource and computation capabilities are typically limited compared to the central cloud servers. The advancement of the MEC platform [10], [11] provides the improvement of scalability, minimized microservice execution delay, and optimal resource utilization over conventional cloud platforms. However, in real life, a dedicated MEC platform is not alone sufficient to support a variety of real-time mobile microservices [12]–[15]. Hence, the edge computing platform was eventually supported by a remote cloud computing platform via the Internet in order to offload some of the applications to remote cloud servers according to their application requirements. On the other hand, the interactions between multiple MEC platforms in order to help the MIoTDs to offload and share their computation microservices with optimized microservice execution delay, maximizing resource utilization, and network throughput [16].

In MEC, a natural problem is deciding the location and time to offload the microservices from MIoTDs. One may offload their computational microservices either to an edge server or remote cloud server based on their application types. However, the MIoTDs need to choose between the edge or remote cloud platform based on their application requirements, such as, microservice execution delay, computational power, and resource requirement. Along with it, the edge and remote cloud servers need to identify, which microservices should be offloaded and served first from the microservice pool. In MEC, the microservices are offloaded while following the First-Come-First-Serve scheme. However, due to the resource-hungry characteristics of the MIoTDs, it is necessary to provide fair resources to them in order to process and offload the heterogeneous microservice-based applications in real-time. In addition, massive connectivity and emerging applications affect the

- *This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. IITP-2023-RS-2022-00156353, Development of End-to-End 8 Ultra-Communication and Networking Technologies). This work was also supported by the RUDN University Strategic Academic Leadership Program (recipient Ammar Muthanna).*
Corresponding authors: Nhu-Ngoc Dao and Sungrae Cho.
- *Amit Samanta is with the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, West Bengal - 721302, India (e-mail: amit.samanta049@gmail.com).*
- *Quoc-Viet Pham is with the School of Computer Science and Statistics, Trinity College Dublin, Dublin 2, D02 PN40, Ireland (e-mail: vietpq@ieee.org).*
- *Nhu-Ngoc Dao is with the Department of Computer Science and Engineering, Sejong University, Seoul 05006, Republic of Korea (e-mail: nndao@sejong.ac.kr).*
- *Ammar Muthanna is with the Peoples' Friendship University of Russia (RUDN University) 6 Miklukho-Maklaya St, Moscow - 117198, Russian Federation and Department of Telecommunication Networks and Data Transmission, The Bonch-Bruevich Saint-PetersburgState University of Telecommunications, 193232 Saint Petersburg, Russia (e-mail: muthanna.asa@spbgut.ru).*
- *Sungrae Cho is with the School of Computer Science and Engineering, Chung-Ang University, Seoul 06974, Republic of Korea (e-mail: srcho@cau.ac.kr).*

data transmission process in MEC, which eventually makes the edge devices* more resource-constrained in the era of IoT [17], [18]. To offload the online microservices, MIoTDs need to set a budget plan for resource allocation (e.g., radio and computing resources) so as to migrate the computational tasks to the edge server. As a result, MIoTDs may decide to process their tasks locally instead of remotely processing if they do not gain enough utility from the edge computing platform. Hence, it is highly requisite to investigate an efficient incentive mechanism to persuade increased participation of MIoTDs and to maximize the achieved profit.

In real life, the computational microservices provided by the mobile edge devices usually have different and resilient resource demand requirements based on several aspects (e.g., poor communication link, lack of optimal demand estimator, external noise). Therefore, the microservice offloading process becomes quite unreliable and eventually affects the network throughput along with the microservice execution delay. Thus, it is extremely important to design an optimal and resilient demand estimation scheme for the proper estimation of resource demand requirements for MIoTDs. The MEC platform utilizes the demand estimation scheme in order to capture the demand-agnostic property of each individual MIoTD, so that their dynamic and even conflicting demands can be profitable for them. In the MEC platform, the demand estimation and incentive microservice offloading mechanism are eventually associated with each other. Here, the resilient demand estimation scheme generally communicates with the incentive microservice offloading process, which inherently influences its design specifications and model. Most likely, if the demand estimation scheme approximates edge devices' demand in a truthful manner and the demand of each device is given the same importance, then the incentive mechanism does not require discriminating them based on their demands.

Due to the paramount significance of encouraging involvement and profit level, we propose an incentive mechanism for microservice offloading in MEC. However, most of the offloading mechanism in the existing literature does not estimate and design any incentives for MIoTDs participating in the offloading mechanism. Here, we consider a situation, where multiple MIoTDs offload their compute services to edge servers to increase the overall performance of the MEC platform. In a practical scenario, there usually exists multiple MIoTDs contending for available resources, who usually offload computational power to edge servers based on their demand requirements that have already been estimated by using an optimal and resilient demand estimator. Hence, in this paper, we concentrate on such a MEC platform where three individual modules, including the MIoTDs, a MEC platform, as well as an optimal and resilient demand estimator synchronized, and desire to design an incentive mechanism that may determine which MIoTD offloads their computational power to edge servers in what price.

With regard to different existing work, [19]–[40], we come up with a novel microservice Incentive Service Offloading (mISO) framework of MEC platform for multiple edge devices, which consists of a resilient demand estimation scheme that takes into consideration of edge devices' various demand. The proposed framework also includes an incentive mechanism that offloads the computational microservices of edge devices to edge servers. More importantly, mISO's incentive mechanism is designed to apply the double-auction mechanism [41]–[43]. Here, both the edge devices and servers participate in the auction mechanism. The primary contributions of this paper are listed below.

- Unlike the existing works, we propose a novel integrated framework for the MEC platform considering the participation of different users (i.e., edge devices), named mISO, comprised of a demand estimation and an incentive mechanism. Such kind of integrated mechanism design is very much more intricate and challenging than modeling them individually, as it acquires the bilateral interaction between the two individual mechanisms.
- mISO incorporates an incentive double auction-based mechanism for MEC, which enables the involvement of both mobile edge users and microservice providers. It also follows several important and essential features, such as, truthfulness, efficient computational complexity, independent rationality, and optimal social welfare.
- The demand estimation approach of mISO takes into consideration the resource-agnostic behavior of mobile edge devices, and provides higher accuracy in results. We validate the efficiency of our mISO framework through various simulation results and evaluate the impact of resource allocation and microservice execution delay.

We organize the paper into the following sections. Section 2 discusses the existing works on MEC and microservices, followed by the problem statement and the system model in Section 3. We formulate a demand-aware incentive mechanism for microservices in Section 4. Section 5 illustrates the simulation results. Finally, we conclude this paper and highlight some promising directions for future research in Section 6.

2 Related Work

2.1 Incentive and Offloading Mechanism at Edge

Due to the paramount importance of minimizing delay for mobile edge devices, the research community has recently developed various incentive and offloading mechanisms for the MEC platform. Gao *et al.* [19] proposed a two-stage algorithm for computation offloading in MEC systems. While the first stage is to decide the offloading ratio of mobile users, the latter stage is to find the optimal processing order for the offloaded tasks, which is solved by an aggregative game approach. In [20], a pricing framework was proposed to allocate a set of heterogeneous computing resources to different services. Both centralized and distributed solutions are proposed, and all the solution approaches can converge to a market equilibrium point. In [21], Zhang *et al.* adopted the cooperation between cloud and edge computing to devise a wholesale-buyback scheme. Similar to [20], two scenarios are considered, one to optimize social welfare and

*. We use MIoTD and edge devices interchangeably in the paper.

Table 1: A synopsis of recent related works on edge computing and microservices

Work	Demand-Agnostic	Incentive	Welfare	Latency	Profit Level
Samanta *et al.* [32], Nguyen *et al.* [20]	✗	✓	✓	✓	✗
Gao *at al.* [19], Zhang *et al.* [21],	✗	✗	✗	✓	✗
Ma *et al.* [22], Wang *et al.* [23], Liu *et al.* [24]	✗	✓	✓	✗	✗
Joseph *et al.* [33], Bao *et al.* [34]	✗	✗	✗	✗	✗
Wang *et al.* [35], Abdullah *et al.* [36], [37]	✗	✗	✗	✓	✓
Our Solution (**mISO**)	✓	✓	✓	✓	✓

the other to optimize the net profit. Another pricing scheme for edge-cloud computing systems can be found in [22], where a resource provisioning algorithm is proposed based on the piecewise convex optimization technique. The use of auction theory for incentive-based resource optimization in MEC systems was investigated in [23]. In [24], the microeconomic theory was employed to design a low-complexity pricing mechanism for MEC systems. In [25], Hou *et al.* adopted a metaheuristic, namely particle swarm optimization, to optimize the reliable computation offloading problem in edge computing and software-defined networking-empowered IoV systems. Applications of game theoretic approaches (matching theory and coalitional game) for computation offloading in MEC systems were considered in [26], [27]. Two problems of edge user allocation were studied in [28]: one to maximize the number of satisfied users and the other to minimize the number of edge servers. Due to the NP-hardness of these problems, Lai *et al.* [28] proposed heuristic approaches to obtain suboptimal solutions with low computational complexity. In [29], the collaborative caching problem in ultra-dense networks was studied, which is solved by the graph coloring method and parallel Gibbs sampling. Li *et al.* [30] used Lyapunov optimization to solve the trusted computation offloading problem in MEC systems. Samanta *et al.* [31] designed a delay-agnostic microservice offloading method for MEC systems to minimize microservice latency.

2.2 Microservice Selection and Resource Allocation

Besides studies on incentive design and offloading decisions, many have focused on the problem of microservice selection and resource allocation in edge computing systems. Samanta *et al.* [32] first considered a scenario in which edge computing resources are not always available and then devised an auction-based scheme to spare the resources allocated to some microservices. Further, an online incentive scheme was proposed to avoid dependency on future bids and requests. The microservices allocation and scheduling problems were studied in [33], [34], in which several heuristic algorithms are proposed to obtain efficient solutions. Exploiting the fact that the collaboration between edge hosts can enhance microservices, a delay-aware coordination problem was considered in [35]. To solve this problem, dynamic programming, and reinforcement learning are used to achieve the optimal solution and online solution, respectively. To better improve the quality of service (QoS) of microservices, [36], [37] proposed predictive autoscaling mechanisms by taking into account the busty particularities of dynamic computational tasks. Another autoscaling method can be found in [38], where Bayesian optimization is used jointly with a heuristic approach to obtain the service scale solution. In [40], the microservice orchestration problem was studied under uncertainty, security, and QoS requirement constraints. The proposed genetic method, namely GA-Par, is shown to outperform a greedy method 42.34% while converging 4x times faster than an existing genetic approach.

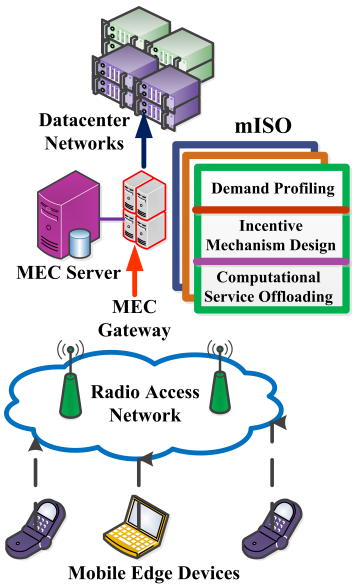


Figure 1: Basic MEC architecture with mISO components.

Synthesis. Unlike most of the aforementioned existing works, which only concentrated on minimizing microservice execution delay and energy consumption for offloading mechanism. These existing works are not deemed to fit for microservice-enabled edge platforms, as microservice have different unique characteristics than normal monolithic services. For easier understanding, we provide a comparative study with different existing works in Table 1. In this work, we come up with a novel double auction-based incentive mechanism for the MEC platform in the presence of heterogeneous edge devices and edge servers, which participate in the process of getting a fair amount of resources. This is one of the preliminary works to design and model an incentive mechanism for the MEC platform. In addition, existing works also did not consider any resilient demand estimation and incentive microservice offloading scheme, which is very tough to model efficiently while maximizing the network performance. Our work considers integrating these two modules, demand estimation and incentive scheme, into

a holistic framework. Here, we consider multiple devices that try to migrate their computational tasks to edge servers based on their demand requirements.

3 System Model

3.1 High-Level Scenario

At a high level, this paper addressed the problem of offloading microservices to edge servers through MEC architecture. The microservice offloading is modeled using the double-auction theory to design an incentive mechanism to provide optimal profits to both the MEC platform and edge users. To design such an intensive mechanism, first, we identify the types of microservices at the edge platforms and then estimate the demands to execute the microservices after offloading to edge servers. Then, we design offloading algorithms to provide better scalability, utilization, and profit level. mISO has two main components – (a) a demand estimation scheme and (b) an incentive approach to provide profit to edge devices. Referring to Fig. 1, the system includes a MEC server located at the MEC gateway that gathers data from all IoT devices. Here, the proposed algorithm is implemented at the MEC gateway as a function. For example, if the MEC gateway is an SDN switch, the proposed algorithm can be deployed as a third-party application running on the controller. To estimate the demand for microservices, we need to know what kind of microservices (critical or non-critical) need to be offloaded to edge servers. To identify that, we propose a microservice identification mechanism described in Section 3.3. Then, we have a demand estimator (in Section 4.1), which is responsible for estimating the resources to offload the microservices to edge servers. After that, we present an incentive approach (in Section 4.2), which provides a profitable microservice offloading mechanism to improve performance in terms of latency and social welfare.

3.2 Problem Statement

Here, we discuss the proposed system model, offloading mechanism, auction model, and main design objectives. We have listed important parameters in Table 2. We design an mISO scheme consisting of a MEC platform, a set of edge devices, $\mathcal{M} = \{M_1, M_2, \cdots, M_K\}$ and a set of edge servers, $\mathcal{F} = \{F_1, F_2, \cdots, F_N\}$. Each MIoTD $M_j \in \mathcal{M}$ has a computational microservice λ_j to be executed by the edge servers. Each of the servers is designed to serve a set of microservices. The set of microservices from all MIoTDs is denoted by $\bar{\lambda} = \{\lambda_1, \lambda_2, \cdots, \lambda_K\}$. The MIoTDs release the real-time microservices following Poisson's distribution in different time instants. Here, we consider t_j as the offloading time of the microservice λ_j. The edge device will dispatch microservice to an edge server immediately after its release. We assume that the servers are not allowed to migrate a microservice from one server to another server in order to overcome the system and migration overhead. We consider a scenario in which each MIoTD is comprised of a set of different binary computational microservices $\mathcal{C}$, where first we need to identify the microservice classes locally, and then we need to offload the microservices to edge servers.

Each microservice λ_j is associated with a unique criticality tag $h_j \in \{+O, -O\}$, which is unknown to the edge devices, the designed platform, and the edge servers. O represents the critically index of microservices, where $+O$ denotes the microservice with maximized resources and $-O$ denotes the microservice with limited resources. If an edge device selects a microservice λ_j to offload it to edge servers, then the edge device will provide a tag h_{ij} to the designed platform. We design a matrix $H = [h_{ij}] \in \{+O, -O, \Theta\}^{N \times M}$, which contains all the labels of edge devices. Here, $h_{ij} = \Theta$ denotes the label to identify that microservice λ_j is not offloaded by edge device M_j. Each device M_j has a different and unique demand $\mathbb{D}_j$ to offload their microservices optimally. For each microservice λ_j, the designed platform accumulates the labels of different edge devices into an accumulated result denoted as $\bar{h}_j$. In Figure 1, we have an overall infrastructure of an edge computing platform with the main components of mISO, and the overall architectural view of mISO is shown in Figure 2. In Figure 2, we have detailed functionalities and components

Table 2: Table of Notations

Parameters	Values
$\mathcal{M}$	Set of edge devices
$\mathcal{F}$	Set of edge servers
$\bar{\lambda}$	Set of microservices
$\mathcal{S}_j$	Offloading cost
β_i	Edge servers bidding price
O	Criticality index of microservices
Θ	Label to identify the microservices
$\mathcal{G}$	Demand of edge devices
$\mathcal{I}$	Capacity of edge servers
$\boldsymbol{\alpha}$	Bidding profile of edge devices
$\boldsymbol{\beta}$	Bidding profile of edge servers
$\mathcal{U}_j^M$	Utility function for edge device
$\mathcal{U}_i^F$	Utility function for edge server
$\mathbb{P}_{lev}$	Profit level
$\mathbb{W}_j$	Offloading decision metric
$Y_{\mathcal{M}}$	Winning edge device set
$Y_{\mathcal{F}}$	Winning edge server set

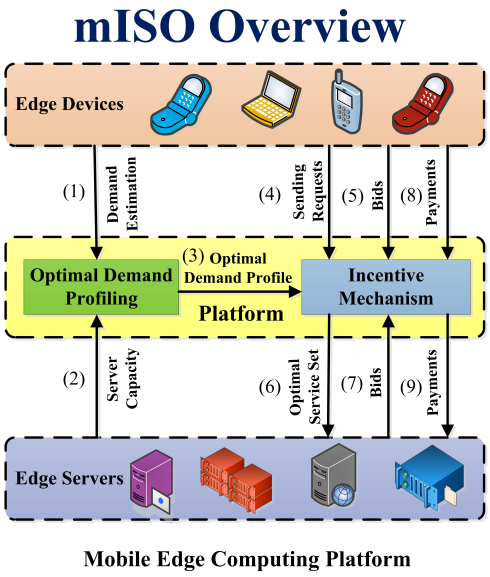


Figure 2: Detailed functionalities of mISO framework and its components.

of mISO, where the numbers define the occurrence of corresponding events. In the figure, the rectangular color boxes represent the different layers, such as light blue representing the edge servers layer, light yellow representing the edge platform layer, and light orange color representing the edge devices layer. They interact with each other through the components in each layer. At first, the demand for edge devices is estimated to know the resource requirements of microservices. Then, the edge devices submit their bids to offload the microservices to edge servers. The incentive mechanism provides a profitable microservice offloading mechanism to improve the overall performance in terms of latency and social welfare.

Resilient Demand Estimation. The demand response of edge devices is very important to estimate the demand profile of different microservices. As the demand response of edge devices changes frequently; therefore, we consider this as demand-agnostic property. The demand response of edge devices is dependent on the packet processing demands and also the capacity of edge servers. The demand of edge devices is denoted as $\mathcal{G} = \{G_1, G_2, \cdots, G_K\}$ and also the capacity of edge servers is denoted as $\mathcal{I} = \{I_1, I_2, \cdots, I_N\}$. We estimate the demand response of edge devices and also the capacity of edge servers in (Section 4.1).

Incentive Mechanism. At first, in a double-auction mechanism, each edge device M_j provides the corresponding microservice λ_j to the designed platform for efficient microservice offloading, and a bid α_j, the amount which is to be paid if a microservice is offloaded efficiently. Thereafter, the designed platform reports the set of microservices $\bar{\lambda}$ to edge servers. After getting the set of microservices $\bar{\lambda}$, each edge server F_i collects the set of microservices from the designed platform in order to execute them, such as, $\gamma_i \in \bar{\lambda}$. The edge servers also submit a bid β_i to execute the microservices efficiently. After receiving the bids, the incentive module in the designed platform decides the set of successful edge devices $Y_{\mathcal{M}}$, set of successful edge servers $Y_{\mathcal{F}}$, the remittance p_j^M charged from each successful edge devices M_j, and the payment p_i^F paid to each successful edge server F_i. The unsuccessful edge devices' microservices do not get executed, hence they do not submit any payment. Likewise, unsuccessful edge servers do not get any payment, as the edge devices do not offload any microservice. Thereafter, the designed platform accumulates the executed microservices submitted by the edge devices and collects the accumulated results. Then, the accumulated results are sent to the successful edge devices for further processing. Here, we consider two different bidding profiles for edge devices and servers denoted as $\boldsymbol{\alpha} = \{\alpha_1, \alpha_2, \cdots \alpha_K\}$ and $\boldsymbol{\beta} = \{\beta_1, \beta_2, \cdots \beta_N\}$. Further, we also consider two payment profiles for edge devices and servers denoted as $\boldsymbol{p}^M = \{p_1^M, p_2^M, \cdots p_N^M\}$ and $\boldsymbol{p}^F = \{p_1^F, p_2^F, \cdots p_K^F\}$.

3.3 Microservice Identification

Before edge devices M_j submit their available microservice λ_j to a designed platform, the microservices are associated with a random variable C_j in order to identify their classes. Here, we consider three kinds of microservice classes - *critical*, *normal*, and *background*†. Hence, we define the microservice label for each microservice of an edge device in Def. 1.

Definition 1. *(Microservice Label). An edge device M_j's microservice label Θ_j of microservice λ_j for edge server F_i is defined as the probability that an edge device is associated with a correct criticality label for a microservice. Mathematically,*

$$\Phi_{ij} = Pr[H_{ij} = h_j] \in [0, 1]. \tag{1}$$

Here, we define a microservice label matrix for edge server denoted as $\boldsymbol{\Phi} = [\Phi_{ij}] \in [0, 1]^{K \times N}$.

Here, the platform could obtain microservice classes by following the discussed approach. If the value of Φ_{ij} is closer to 1, i.e., $\Phi_{ij} \approx 1$, then the microservice class is associated with the critical microservice class. Similarly, if the value of Φ_{ij} is closer to 0.5, i.e., $\Phi_{ij} \approx 0.5$, then the microservice class is associated with the normal microservice class; otherwise, it is associated to the background microservice class. We modeled our platform in a way that after edge devices submit their microservices to the platform, the platform perceives the microservice label matrix a priori and also stores the previous values in the record. In practice, edge devices tend to have similar microservice labels for similar kind of microservices, the platform provides this information to edge servers for different microservices with different unique microservice labels, and consider edge devices' identities to calculate their microservice labels of similar microservices.

3.4 Double-Auction Mechanism

In this paper, we set to design a framework where both the edge devices and servers are considered to be strategic, which focuses on the maximization of their individual utility values. mISO considers a double-auction mechanism, where both the edge devices and microservices are involved in this process. We employ the double-auction mechanism, as defined in the following definition.

Definition 2. *For the double-auction mechanism of our MEC platform, edge device M_j gets a value R_j if a microservice λ_j is successfully offloaded and bids to the designed platform α_j. α_j is the amount that the edge device is willing to pay for a successful microservice offloading. Each edge server F_i is interested in offloading one subset of the microservices $\gamma_i \in \bar{\lambda}$, and bids to the designed platform β_i. β_i is the edge server bidding price for offloading all the microservices. The real offloading cost for offloading to the servers γ_i is denoted as $\mathcal{S}_i$ (calculated using the formulation in [18]). Here, in mISO, both the edge devices' and servers' bids are hidden from the designed platform.*

Here, we consider the utility functions for edge devices and servers. Also, we define the profit level of a designed platform in Def. 3, 4, and 5.

†. Some of the examples of different microservices: critical (i.e., augmented reality (AR)), normal (general sensor data), and background (updates).

Definition 3. *(Device Utility). The utility function $\mathcal{U}_j^M$ for an edge device M_j is defined as:*

$$\mathcal{U}_j^M = \begin{cases} R_j - p_j^M, & M_j \in Y_{\mathcal{M}}, \\ 0, & otherwise. \end{cases} \tag{2}$$

Definition 4. *(Server Utility). The utility function $\mathcal{U}_i^F$ for an edge server F_i is defined as:*

$$\mathcal{U}_i^F = \begin{cases} p_i^F - \mathcal{S}_i, & F_i \in Y_{\mathcal{F}}, \\ 0, & otherwise. \end{cases} \tag{3}$$

Definition 5. *(MEC Platform Profit). The profit level for the designed MEC platform is defined as:*

$$\mathbb{P}_{lev} = \sum_{j:M_j \in Y_{\mathcal{M}}} p_j^M - \sum_{i:F_i \in Y_{\mathcal{F}}} p_i^F. \tag{4}$$

Depending on the above definitions, we formulate the social welfare of MEC platform.

Definition 6. *The social welfare of the MEC platform is expressed as:*

$$\mathbb{S}_{wel} = \mathbb{P}_{lev} + \sum_{j:M_j \in \mathcal{M}} \mathcal{U}_j^M + \sum_{i:F_i \in \mathcal{F}} \mathcal{U}_i^F \tag{5}$$

$$= \sum_{j:M_j \in \mathcal{M}} R_j - \sum_{i:F_i \in \mathcal{F}} \mathcal{S}_i. \tag{6}$$

Here, social welfare is defined as the summation of the designed MEC platform's profit levels and all edge devices' and servers' utility values.

3.5 Design Goals

Here, we design mISO in such a way that it carries the following beneficial properties. As the edge devices are selfish and strategic for mISO, hence it is considered that every edge device M_j provides a bid α_j that differs from R_j, which is the actual value for the microservice λ_j. Furthermore, edge server F_i may also send a bid β_i that is different from $\mathcal{S}_i$ which is the total cost of offloading and executing all the microservices γ_i. Hence, the main goal is to construct an incentive mechanism explained in Def. 7.

Definition 7. *The mISO double-auction mechanism can be considered to be truthful as long as the bidding R_j and $\mathcal{S}_i$ is a dominant strategy for edge device M_j and server F_i, i.e., the bidding strategies R_j and $\mathcal{S}_i$, maximize the utility of edge device M_j and server F_i, nevertheless of other edge devices' and servers' bids.*

Using the formation definition, we focus on securing truthful bids to the designed platform from both edge devices and servers. Along with truthfulness, another important goal is individual rationality, as discussed in Def. 8.

Definition 8. *The mISO double-auction mechanism is considered to be individually rational, if and only if none of the edge devices or servers acquire any kind of negative utility values, i.e., $\mathcal{U}_j^M \geq 0$, and $\mathcal{U}_i^F \geq 0$, for each edge device M_j and server F_i, respectively.*

To encourage the participation of both the edge devices and servers in the auction mechanism, individual rationality is a powerful property as it guarantees that the bill to an edge device is smaller than its submitted bid, and an edge server's offloading price is also articulately remunerated. As the edge servers aggregate their microservice labels to secure the accumulated results so that they can be considered to be accurate in the edge platform, discussed in Def. 9.

Definition 9. *A microservice λ_j is offloaded with $\mathbb{K}_j$-accuracy given the condition $Pr[\bar{H}_j \neq h_j] \leq \mathbb{K}_j$, where $\mathbb{K}_j \in (0, 1)$, and $\bar{H}_j$ defines the arbitrary value denoting the accumulated outcome for microservice λ_j.*

From the Def. 9, $\mathbb{K}_j$-accuracy provides the accumulated outcome equivalent to the true microservice label with the higher probabilistic value. For each edge microservice, λ_j, $\mathbb{K}_j$ is a parameter selected by the designed platform, and a small value of $\mathbb{K}_j$ signifies an important specification of the accuracy level. Briefly, here the main goal is to provide adequate microservice labels to edge servers with higher accuracy for all offloaded edge microservices and also encourage involvement in the incentive mechanism for both edge devices and servers while considering individual rationality and truthfulness.

4 Mathematical Framework

In this section, mathematical models for the resilient demand and incentive mechanism of mISO are discussed.

4.1 Resilient Demand Estimation

This section introduces a resilient demand estimation approach and also analyzes its theoretical aspects.

4.1.1 Proposed Approach

The resource demand vector for edge device M_j is obtained by multiplying its bandwidth demand (in packets/s) by per-packet processing demands (in resource/packet) and is denoted as $< D_{R_1,V_1}, D_{R_2,V_2}, \cdots, D_{R_j,V_j} >$, where R_j denotes the bandwidth demand of microservice λ_j and V_j denotes the processing demand of microservice λ_j.

Definition 10. *The demand response G_j of edge device M_j is mathematically expressed as:*

$$G_j = \sum_{q=1}^{\mathbb{F}} \frac{h_{R_j} D_{R_j,V_j} Z_{f_q} \mathcal{L}_{f_q}}{\mathbb{C}_{f_q}}. \tag{7}$$

where h_{R_j} denotes the resource flow for microservice λ_j, Z_{f_q} denotes the microservice flow constraint, $\mathcal{L}_{f_q}$ denotes the weight of a microservice flow f_q, and $\mathbb{C}_{f_q}$ denotes the total resource capacity of a microservice flow f_q.

The capacity vector for edge server F_i is defined as $< I_1, I_2, \cdots, I_F >$. The capacity of a server is depended on the initial load of the server and the total allocated bandwidth to the virtual machine (VM) for in and out microservice flows.

Definition 11. *The capacity I_i of an edge server F_i is mathematically defined as:*

$$I_i = \mathcal{L}_{ini} + \left(\frac{\eta \mathcal{B}_i^{in} + \zeta \mathcal{B}_i^{out}}{\mathcal{B}_i^{tot}} \right), \tag{8}$$

where $\mathcal{L}_{ini}$ denotes the initial load of an edge server, $\mathcal{B}_i^{in}$ and $\mathcal{B}_i^{out}$ denote the bandwidth allocated to VMs for inflow and outflow

microservices, η and ζ denote the binary factor for bandwidth allocated to VMs for inflow and outflow microservices, and $\mathcal{B}_i^{tot}$ denotes the total bandwidth available for server F_i.

Definition 12. *The offloading decision metric $\mathbb{W}_j$ for an edge device M_j is defined as:*

$$\mathbb{W}_j(I_i, \mathcal{E}_j) = \frac{1}{\mathcal{E}_j}\left(I_i - \frac{\mathcal{A}_j \chi_j f_j}{\mathcal{J}_j^{cap}}\right), \tag{9}$$

where $\mathcal{A}_j$ denotes the microservice flow constraint for microservice λ_j, χ_j denotes the microservice flow rate for microservice λ_j, $\mathcal{J}_j^{cap}$ denotes the capacity of an edge device M_j, $\mathcal{E}_j$ denotes the offloading capacity and f_j denotes the dedicated flows.

Hence, we have designed an optimization problem for efficient microservice offloading in MEC. Mathematically,

$$\max \sum_{j=1}^{K} \mathbb{W}_j(I_i, \mathcal{E}_j) = \sum_{j=1}^{K}\sum_{i=1}^{N} \frac{1}{\mathcal{E}_j}\left(I_i - \frac{\mathcal{A}_j \chi_j f_j}{\mathcal{J}_j^{cap}}\right) \tag{10}$$

$$\text{subject to} \quad I_i < I_{th}, i \in N \tag{11}$$

$$\mathcal{E}_j \geq \mathcal{E}_{th}, j \in K \tag{12}$$

$$\chi_j \geq \mathcal{S}_{th}, j \in K. \tag{13}$$

Here, Eq. (11) implies that a load of edge server I_i is to be less than the threshold load of an edge server I_{th}. The offloading capacity of edge device $\mathcal{E}_j$ is required to be larger than the threshold offloading capacity $\mathcal{E}_{th}$, as indicated in Eq. (12). Eq. (11) represents that the microservice flow rate χ_j needs to be higher than the threshold microservice flow rate‡ $\mathcal{S}_{th}$. Applying the duality technique to solve the problem (13), the Lagrangian function is mathematically given as follows:

$$\mathbb{L}_j = \sum_{j=1}^{N} \frac{\bigtriangledown_j}{\mathbb{W}_{th}(I_i, \mathcal{E}_j)} \mathcal{L}_j(I_i, \mathcal{E}_j) - \mathbb{A}_1\Big(\sum_{i=1}^{K} I_i - I_{th}\Big) - \mathbb{A}_2\Big(\sum_{j=1}^{N} \mathcal{E}_j - \mathcal{E}_{th}\Big) - \mathbb{A}_3\Big(\sum_{j=1}^{N} \chi_j - \mathcal{S}_{th}\Big), \tag{14}$$

where $\mathbb{A}_1$, $\mathbb{A}_2$ and $\mathbb{A}_3$ are the Lagrangian dual variables and $\bigtriangledown_j$ indicates the priority index of edge devices based on the application types.

4.1.2 Theoretical Analysis

Here, we theoretically analyze the offloading decision metric for MEC.

Theorem 1. *The offloading metric, $\mathbb{W}_j(I_i, \mathcal{E}_j)$, in Definition (12) is continuous over the interval $0 < \mathcal{E}_j < \mathcal{E}_{max}$.*

Proof. Let, the offloading metric, $\mathbb{W}_j(I_i, \mathcal{E}_j)$, a real valued function designed on a subset of, ξ, of real number, $\mathbb{R}$. Specifically, $\xi \in (\psi, \mathcal{E}_{max})$, where $\psi > 0$. We assume that there exists a $\xi > 0$, such that for all $\mathcal{E}_j \in \xi$, and $\delta_0 \in \xi$, the inequality $|\mathcal{E}_j - \delta_0| < \xi$. Therefore, we get,

$$|\mathbb{W}_j(I_i, \mathcal{E}_j) - \mathbb{W}_j(I_i, \delta_0)| = \left|\frac{\delta_0 - \mathcal{E}_j}{\mathcal{E}_j \delta_0}\left(I_i - \frac{\mathcal{A}_j \chi_j f_j}{\mathcal{J}_j^{cap}}\right)\right|$$

‡. As the microservices are comprised of small tasks, therefore there has data dependency among tasks hence we consider a data flow rate between the tasks.

$$\Rightarrow \quad |\mathbb{W}_j(I_i, \mathcal{E}_j) - \mathbb{W}_j(I_i, \delta_0)| = \left|\frac{\delta_0 - \mathcal{E}_j}{\mathcal{E}_j \delta_0}\left(I_i - \frac{\mathcal{A}_j \chi_j f_j}{\mathcal{J}_j^{cap}}\right)\right|$$

$$\Rightarrow |\mathbb{W}_j(I_i, \mathcal{E}_j) - \mathbb{W}_j(I_i, \delta_0)| = \Xi \tag{15}$$

where $\Xi = \frac{\delta_0 - \mathcal{E}_j}{\mathcal{E}_j \delta_0}\big(I_i - \frac{\mathcal{A}_j \chi_j f_j}{\mathcal{J}_j^{cap}}\big)$. As $\mathcal{E}_j \in (\psi, \mathcal{E}_{\max})$, $\forall j \in (1, N)$, all the terms in Eq. (15) are positive. Hence, $\Xi > 0$. Therefore, we conclude that $\mathbb{W}_j(I_i, \mathcal{E}_j)$ is a continuous function over the range $(\psi, \mathcal{E}_{\max})$, where $\psi > 0$. Hence, the proof concludes. □

Lemma 1. *The offloading mechanism is loss-less in nature for edge devices in the MEC platform.*

Proof. The offloading mechanism is lossless in nature. To prove that, we have introduced a loss-less metric as:

$$\sum M_{j'} \in \hat{\mathcal{M}} \mid \exists M_{j''} \in \bar{\mathcal{M}}, \mathbb{W}_{j'}(I_i, \mathcal{E}_j) > \mathbb{W}_{j''}(I_i, \mathcal{E}_j). \tag{16}$$

where $\hat{\mathcal{M}}$ and $\bar{\mathcal{M}}$ are the set of critical edge devices and the set of the remaining edge devices of the maximal subset, respectively, $\hat{\mathcal{M}} \cup \bar{\mathcal{M}} = \mathcal{M}$. We use the method of contradiction to prove the statement. We consider $\exists M_j, M_{j''}$ in which Eq. (16) holds true. Thus, ($\mathbb{L}_{j'} > \mathbb{L}_{j''}$). We have,

$$\sum_{j'=1}^{\mathcal{M}-\{\mathcal{M}_k\}+\{\mathcal{M}_l\}} \mathbb{L}_{j'}(I_i, \mathcal{E}_j) > \sum_{j''=1}^{\mathcal{M}} \mathbb{L}_{j''}(I_i, \mathcal{E}_j). \tag{17}$$

However, $\mathbb{L}(I_i, \mathcal{E}_j)$ is maximized, where $\forall \mathcal{M}_k \in \hat{\mathcal{M}}, \mathcal{M}_l \in \bar{\mathcal{M}}$. Thus,

$$\sum_{k'=1}^{K} \bigtriangledown_{k'} \frac{\sum_{j'=1}^{\mathcal{M}-\{\mathcal{M}_k\}+\{\mathcal{M}_l\}} \mathbb{L}_{j'}(I_i, \mathcal{E}_j)}{\mathcal{U}_{th}(S, \mathcal{Z}_i)} > \sum_{k''=1}^{K} \bigtriangledown_{k''} \frac{\sum_{j''=1}^{\mathcal{M}} \mathbb{L}_{j''}(I_i, \mathcal{E}_j)}{\mathcal{U}_{th}(S, \mathcal{Z}_i)}. \tag{18}$$

Derived from Eq. (18), we have,

$$\sum_{j'=1}^{\bar{K}} \mathbb{L}_{j'}(I_i, \mathcal{E}_{j'}) > \sum_{j''=1}^{K} \mathbb{L}_{j''}(I_i, \mathcal{E}_{j''}). \tag{19}$$

In other words, mathematically, we get,

$$\sum_{j'=1}^{\bar{K}} \mathcal{E}_{j'} > \sum_{j''=1}^{K} \mathcal{E}_{j''} \tag{20}$$

$$\Rightarrow \sum_{j'=1}^{\bar{K}} \mathbb{W}_{j'}(I_i, \mathcal{E}_{j'}) > \sum_{j''=1}^{K} \mathbb{W}_{j''}(I_i, \mathcal{E}_{j''}). \tag{21}$$

Thus, the offloading decision metric $\nexists \mathbb{W}_{j'}(I_i, \mathcal{E}_{j'}) > \mathbb{W}_{j''}(I_i, \mathcal{E}_{j''})$ thereby disproving our assumption. The offloading mechanism is lossless in nature for edge devices. Hence, the proof concludes. □

Theorem 2. *The online offloading mechanism is competitive in nature for real-time edge applications.*

Proof. To show competitiveness, we have considered the dual technique to solve the original problem. To analyze the online algorithms [44], the dual fitting and main-dual approach techniques have been modeled. We introduce dual variables $\mathbb{A}_1$, $\mathbb{A}_2$ and $\mathbb{A}_3$ for the first three constraints in

Eq. (10). Let $\mathcal{W}$ denote the set of $\mathbb{W}_j(I_i, \mathcal{E}_j)$ that satisfy the three constraints Eq. (10). However, using the dual function, we have,

$$\mathbb{L}(\mathbb{A}_1, \mathbb{A}_2, \mathbb{A}_3) = \sum_{j=1}^{N} \frac{\bigtriangledown_j \mathcal{L}_j(I_i, \mathcal{E}_j)}{\mathbb{W}_{th}(I_i, \mathcal{E}_j)} - \mathbb{A}_1\Big(\sum_{i=1}^{K} I_i - I_{th}\Big) - \mathbb{A}_2\Big(\sum_{j=1}^{N} \mathcal{E}_j - \mathcal{E}_{th}\Big) - \mathbb{A}_3\Big(\sum_{j=1}^{N} \chi_j - \mathcal{S}_{th}\Big), \tag{22}$$

$$\mathbb{L}'(\mathbb{A}_1, \mathbb{A}_2, \mathbb{A}_3) = \max \sum_{\mathbb{W}_j \in \bar{\mathbb{W}}} \left(\sum_{j=1}^{N} \frac{\bigtriangledown_j \mathcal{L}_j(I_i, \mathcal{E}_j)}{\mathbb{W}_{th}(I_i, \mathcal{E}_j)}\right) - \mathbb{A}_1\Big(\sum_{i=1}^{K} I_i - I_{th}\Big) - \mathbb{A}_2\Big(\sum_{j=1}^{N} \mathcal{E}_j - \mathcal{E}_{th}\Big) - \mathbb{A}_3\Big(\sum_{j=1}^{N} \chi_j - \mathcal{S}_{th}\Big). \tag{23}$$

where $\bar{\mathbb{W}}$ denotes the set of offloading decision metric and $\mathbb{L}'(\mathbb{A}_1, \mathbb{A}_2, \mathbb{A}_3)$ denotes the maximum value of $\mathbb{L}(\mathbb{A}_1, \mathbb{A}_2, \mathbb{A}_3)$. As stated in [45], we used the weak duality theorem, the function of the weak duality theorem capitulates a maximum limit on the optimal solution of the main proposed problem for any $\mathbb{A}_1 \geq 0$, $\mathbb{A}_2 \geq 0$, and $\mathbb{A}_3 \geq 0$. The main idea of the dual fitting is to set dual variables $(\mathbb{A}_1, \mathbb{A}_2, \mathbb{A}_3)$ based on the values of the primal variables $\bar{\mathbb{W}}$ resolved by an online approach with deterministic value such that $\mathbb{L}'(\mathbb{A}_1, \mathbb{A}_2, \mathbb{A}_3) \leq \Lambda \bar{F}(\bar{\mathbb{W}})$ for some $\Lambda \geq 1$, which expresses that this online approach is competitive. The proof ends. □

4.2 Incentive Mechanism for Microservice Offloading

We discuss the design specifications of mISO's incentive mechanism considering its mathematical model. We also prove the NP-hardness for the proposed approach, while providing the necessary theoretical analysis.

4.2.1 Design Details of Microservice Offloader

As discussed in Section 3.4, the incentive mechanism for mISO is described in Def. 2. Here, we envision modeling an optimal double auction mechanism, which maximizes social welfare and also provides satisfactory demand for edge devices.

Social-Welfare Maximization. We mathematically defined the winner selection scheme through a double-auction social-welfare maximization approach. Mathematically,

$$\max \sum_{j:\lambda_j \in \bar{\lambda}} \alpha_j \Phi_j - \sum_{i:F_i \in \mathcal{F}} \beta_i \Psi_i, \tag{24}$$

$$\text{subject to} \quad G_j \leq G_{th}, M_j \in \mathcal{M} \tag{25}$$

$$\Psi_i, \Phi_j \in \{0, 1\}, \forall F_i \in \mathcal{F}, \lambda_j \in \bar{\lambda} \tag{26}$$

$$\mathcal{E}_j \geq \mathcal{E}_{th}, j \in K. \tag{27}$$

$$\mathbb{W}_j > \mathbb{W}_{th}, M_j \in \mathcal{M} \tag{28}$$

Constants. Here, mISO captures several parameters as inputs: set of microservices $\bar{\lambda}$, set of edge servers $\mathcal{F}$, bidding profile of edge devices' and servers' $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, concerned microservice profile set of edge servers' $\boldsymbol{\gamma} = \{\gamma_1, \gamma_2, \cdots, \gamma_N\}$, microservice label matrix H, and $\bar{\mathbb{K}}$ vector.

Variables. Here, $\boldsymbol{\Phi} = (\Phi_1, \Phi_2, \cdots, \Phi_N)$ denotes the binary factors for incentive mechanism. Here $\Phi_i = 1$ denotes that microservice λ_i will be offloaded and executed, and thus, edge device M_j is a winning edge device (i.e., $M_j \in Y_M$), whereas $\Phi_i = 0$ denotes $M_j \notin Y_M$. Similarly, the mechanism considers another vector of K binary variables, $\boldsymbol{\Psi} = (\Psi_1, \Psi_2, \cdots, \Psi_K)$, where $\Psi_i = 1$ indicates that edge server F_i is a winning edge server (i.e., $F_i \in Y_F$), and $\Psi_i = 0$ means $F_i \notin Y_F$.

Main Function. The designed main function satisfies that $\sum_{j:\lambda_j \in \bar{\lambda}} \alpha_j \Phi_j - \sum_{i:F_i \in \mathcal{F}} \beta_i \Psi_i = \sum_{j:M_j \in Y_M} \alpha_j \Phi_j - \sum_{i:F_i \in Y_F} \beta_i \Psi_i$ it is basically the social welfare discussed in Def. 6 considering the edge devices' and servers' bids.

Constraints. Eq. (24) describes the main objective function and Eq. (25) represents that the demand response of an edge device G_j is to be greater than the threshold demand response G_{th}. The offloading decision metric $\mathbb{W}_j$ is to be greater than the threshold offloading metric $\mathbb{W}_{th}$ as shown in Eq. (26). The offloading capacity of edge device $\mathcal{E}_j$ is to be greater than the threshold offloading capacity $\mathcal{E}_{th}$ as shown in Eq. (27).

Theorem 3. *The incentive mechanism is NP-hard in nature.*

Proof. For the designed incentive mechanism, both the parameters α_j and β_i are constant. As both the parameters are constant, hence the mISO incentive mechanism has identical computational complexity in comparison with a special case of the mISO incentive mechanism. The special case of the incentive mechanism is considered to be a binary linear program for both the parameters α_j and β_i. Thus, we prove the NP-hardness of the binary linear program by a polynomial-time reduction from the minimum weight set cover problem. We consider an NP-complete minimum weight set cover problem with a domain of K elements $\bar{\lambda} = \{\lambda_1, \lambda_2, \cdots, \lambda_K\}$ and a domain of N subsets $\boldsymbol{\gamma} = \{\gamma_1, \gamma_2, \cdots, \gamma_N\}$ in order to start the reduction. For each domain $\gamma_i \in \boldsymbol{\gamma}$ has positive weights for both the parameters α_j and β_i. The main concern for the minimum weight set cover problem is to search a subset of $\boldsymbol{\gamma}$ with the minimum weight value. The subset equivalents to the unions of $\bar{\lambda}$. Next, we modify γ_i to γ_i' where each element $\lambda_i \in \gamma_j$ has $\Gamma_{i,j} \in \beth^+$ samples and essential to cover $\mathbb{A}_j \in \beth^+$ instants for each element λ_j. Then, for an instance of the binary linear program with $\boldsymbol{E} = [\Gamma_{i,j}] \in (\beth^+)^{K \times N}$, $\mathbb{E} = [\mathbb{A}_j] \in (\beth^+)^{K \times 1}$, and payment profiles for edge devices and servers $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}$ is generated. Hence, the main problem is illustrated by the binary linear program through the elements in $\boldsymbol{E}$ and $\mathbb{E}$. The values of $\boldsymbol{E}$ and $\mathbb{E}$ are considered to be positive real numbers besides positive integers. Therefore, the mISO incentive mechanism is polynomial-time reducible to the binary linear program, and it proves the NP-hardness. Moreover, as the binary linear program is a special case of the mISO incentive mechanism, thus mISO incentive mechanism is also NP-hard. □

4.2.2 Offloading Algorithm Design

According to Theorem 3, the optimization problem is shown to be NP-hard. It also indicates that there is a need for an efficient algorithm, which provides a guaranteed approximation ratio. Hence, we provide an mISO incentive

Algorithm 1: Winner Selection for Incentive Mechanism

Input: $\bar{\lambda}$, $\mathcal{M}$, $\mathcal{F}$, $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, R_j, $\boldsymbol{\gamma}$, $\mathbb{G}$.
Output: $Y_{\mathcal{M}}$, $Y_{\mathcal{F}}$, $\mathbb{C}$
$Y_{\mathcal{M}} \leftarrow \phi$ and $Y_{\mathcal{F}} \leftarrow \phi$;
$\mathbb{C} \leftarrow OFC(\bar{\lambda}, \boldsymbol{\gamma}, \mathbb{G})$;
for *each* j *s.t.* $\lambda_j \in \bar{\lambda}$ **do**
  $\mathbb{C}_j \leftarrow \{F_i | F_i \in \mathbb{C}, \lambda_j \in \gamma_i\}$;
while $max_{j:M_j \in \mathcal{M}}(\alpha_j - \sum_{i:F_i \in \mathbb{C}_j} \beta_i) \geq 0$ **do**
  $j^* \leftarrow \arg\max_{j:M_j \in \mathcal{M}}(\alpha_j - \sum_{i:F_i \in \mathbb{C}_j} \beta_i)$;
  each device M gets a value $R_j = 1$;
  $Y_{\mathcal{M}} \leftarrow Y_{\mathcal{M}} \cup \{M_j^*\}$;
  $\mathcal{M} \leftarrow \mathcal{M}/\{M_j^*\}$;
  $Y_{\mathcal{F}} \leftarrow Y_{\mathcal{F}} \cup \mathbb{C}_j$;
  if *each* j *s.t.* $M_j \in \mathcal{M}$ *and* $R_j == 1$ **then**
    $\mathbb{C}_j \leftarrow \mathbb{C}_j / \mathbb{C}_j^*$;
Return $Y_{\mathcal{M}}$, $Y_{\mathcal{F}}$;

Algorithm 2: Pricing for Double-Auction Mechanism

Input: $\bar{\lambda}$, $\mathcal{M}$, $\mathcal{F}$, $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$, R_j, $\mathbb{G}$, $Y_{\mathcal{M}}$, $Y_{\mathcal{F}}$, S_i, $\mathbb{C}$
. **Output:** $\boldsymbol{p}^M$, $\boldsymbol{p}^F$
$\boldsymbol{p}^M \leftarrow 0$, $\boldsymbol{p}^F \leftarrow 0$;
for *each* j *s.t.* $M_j \in Y_{\mathcal{M}}$ **do**
  run Alg. 1 on $\mathcal{M}/\{M_j\}$ and $\mathcal{F}$;
  $Y'_{\mathcal{M}} \leftarrow$ winning edge device set after line 3;
  for *each* k *s.t.* $M_k \in Y'_{\mathcal{M}}$ **do**
    $p_j^M \leftarrow \min\{p_j^M, \sum_{F_i \in \mathbb{C}'_j} \beta_i + \alpha_k - \sum_{F_i \in \mathbb{C}'_k} \beta_i\}$;
    if $\mathbb{C}'_j = \phi$ **then**
      $p_j^M \leftarrow \min\{p_j^M, 0\}$;
for *each* i *s.t.* $F_i \in Y_{\mathcal{F}}$ **do**
  run Alg. 1 on $\mathcal{M}$ and $\mathcal{F}/\{F_i\}$;
  calculate the offloading cost S_i;
  $Y'_{\mathcal{F}} \leftarrow$ winning edge server set after line 10;
  for *each* k *s.t.* $F_i \in \mathbb{C}'_k$ *and* $M_k \in Y'_{\mathcal{M}}$ **do**
    sort edge devices $\alpha_j - \sum_{i:F_i \in \mathbb{C}'_j} \beta_i$ in decreasing order;
    $\Omega \leftarrow$ index of the first edge server with $F_i \notin \mathbb{C}'_\Omega$;
    if $F_\Omega \in Y'_{\mathcal{F}}$ **then**
      $p_i^F \leftarrow \max\{p_i^F, \alpha_k - \sum_{F_h \in \mathbb{C}'_k} \beta_h + \alpha_\Omega - \sum_{F_h \in \mathbb{C}'_\Omega} \beta_h\}$;
      get the optimized offloading cost S_i;
    else
      $p_i^F \leftarrow \max\{p_i^F, \alpha_k - \sum_{F_h \in \mathbb{C}'_k} \beta_h\}$;
Return $\boldsymbol{p}^M$, $\boldsymbol{p}^F$;

mechanism with a possible approximation ratio, and the algorithm also guarantees positive social welfare for the edge devices and servers.

We discuss the incentive winner selection algorithm for MEC in Alg. 1. The inputs of the Alg. 1 are a set of edge devices $\mathcal{M}$, a set of edge servers $\mathcal{F}$, a set of microservices $\bar{\lambda}$, bidding profile of edge devices $\boldsymbol{\alpha}$ and servers $\boldsymbol{\beta}$, set of concerned microservices for edge servers $\boldsymbol{\gamma}$, demand profile of edge devices $\mathbb{G}$. First, we initialize the set of edge devices $\mathcal{M}$ and $\mathcal{F}$ servers to ψ. Thereafter, we calculate the optimal feasible cover set (OFC)[§] $\mathbb{C}$, which considers the set of edge servers to provide the feasibility of constraint (25) for each microservice λ_i. We call another function OFC to provide the feasible cover, it takes the inputs: set of microservices $\bar{\lambda}$, set of concerned microservices for edge servers $\boldsymbol{\gamma}$, and demand profile of edge devices $\mathbb{G}$. We implemented the function in polynomial time for edge devices and servers. As an example, the edge server F_i gets selected into the optimal feasible cover set by the OFC function based on the decreasing value of $\sum_{j:\lambda_j \in \gamma_i} G_j$, while satisfying all the constraints.

The complexity of the OFC function is $O(N)$. Here, the OFC function considers a greedy approach in this work. The selection of a particular OFC is not important, since we get a feasible cover in polynomial time. Thereafter, we choose a feasible cover for the set of edge server $\mathbb{C}_j$ for each microservice λ_i whose concerned server sets include this microservice. Depending on the value of $\mathbb{C}$, we select the set of winning edge devices and servers which assign positive social welfare. The process gets terminated till it exceeds $\arg\max_{j:M_j \in \mathcal{M}}(\alpha_j - \sum_{i:F_i \in \mathbb{C}_j} \beta_i)$, the maximum social welfare of considering a new edge device M_j and the set of edge servers $\mathbb{C}_j$, and the winning edge devices and server set get non-positive values, respectively. Each device M gets a value $R_j = 1$. For each step, we find the very first index $j*$ of the edge device M_j* which gives maximum social welfare. Then, we incorporate M_j* into the winning edge device set $Y_{\mathcal{M}}$, also discard M_j* from the edge device set $\mathcal{M}$, and also incorporates all edge servers in $\mathbb{C}_j*$ into the winning edge server set $Y_{\mathcal{F}}$. At last, it discards all the edge servers in $\mathbb{C}_j*$ from $\mathbb{C}_j$ for each microservice λ_j. Then, it returns the winning edge device and server set $Y_{\mathcal{M}}$ and $Y_{\mathcal{F}}$.

Now, we discuss the pricing policy for the double-auction mechanism in Alg. 2. It takes the inputs of Alg. 1 and also takes a few other inputs: winning set of edge devices $Y_{\mathcal{M}}$ and servers $Y_{\mathcal{F}}$. At first, we initialize the payment vector of edge devices and servers to 0. Thereafter, we estimate the payment p_j^M of every edge device. Then, we perform on the edge device set $\mathcal{M}$ and server set $\mathcal{F}$ excluding device M_j, for each $M_j \in Y_{\mathcal{M}}$. We also set the winning edge device set $Y'_{\mathcal{M}}$. We also then find the minimum bid value $\alpha_{j,k}$ for edge device M_j to declare M_k as the winner. This step continue for each $M_k \in Y'_{\mathcal{M}}$. To get that, $\alpha_{j,k}$ should fulfill a condition $\alpha_{j,k} - \sum_{F_i \in \mathbb{C}'_j} \beta_i = \alpha_k - \sum_{F_i \in \mathbb{C}'_k} \beta_i$, it can be identical to $\alpha_{j,k} - \sum_{F_i \in \mathbb{C}'_j} \beta_i + \alpha_k - \sum_{F_i \in \mathbb{C}'_k} \beta_i$. Also, we get the optimized offloading cost S_i for each j s.t. $\lambda_j \in \bar{\lambda}$ using Lagrangian multipliers.

The optimal solution of our scheme can be found using a feasible solution where we get the maximum values of $Y_{\mathcal{M}}$ and $Y_{\mathcal{F}}$. We get a global optimal solution when there are no other feasible solutions with values than $Y_{\mathcal{M}}$ and $Y_{\mathcal{F}}$.

Here, $\mathbb{C}'_1, \mathbb{C}'_2, \cdots \mathbb{C}'_K$ denote the sets $\mathbb{C}_1, \mathbb{C}_2, \cdots \mathbb{C}_K$ when a particular edge device M_k is chosen into $Y'_{\mathcal{M}}$. When $\mathbb{C}'_j$ is non-empty, then the payment p_j^M is set based on the minimum value of $\alpha_{j,k}$'s, otherwise, it should be compared to 0. As the edge device, M_j can win, until it has a positive bid. Then, we estimate the payment p_i^F for each winning edge server. We perform on the edge device set $\mathcal{M}$ and server set $\mathcal{F}$ excluding device F_i, for each $F_i \in Y_{\mathcal{F}}$. Next, we also set the winning edge device set $Y'_{\mathcal{F}}$. Here, $\mathbb{C}'_1, \mathbb{C}'_2, \cdots \mathbb{C}'_K$ denote the sets $\mathbb{C}_1, \mathbb{C}_2, \cdots \mathbb{C}_K$ when a particular edge device M_k is chosen into $Y'_{\mathcal{M}}$. It also estimates the maximum bid $\beta_{i,k}$ for winning edge server F_i for each set $\mathbb{C}'_k$, $\forall F_i \in \mathbb{C}'_k$. We sort the edge servers' social welfare in decreasing order based on the value of $\alpha_j - \sum_{F_i \in \mathbb{C}'_k} \beta_i$ and also discover the index Ω of the very first edge server

§. The optimal feasible cover set in the proposed formulation is a feasible solution that covers all elements to provide the optimal solution.

such as way that F_i is not related to $\mathbb{C}'_\Omega$. When F_Ω i a winning edge server in $Y'_{\mathcal{F}}$, then $\beta_{i,k}$ must follow a condition $\alpha_k - (\sum_{F_h \in \mathbb{C}'_k} \beta_h + \beta_{i,k}) = \alpha_f - \sum_{F_h \in \mathbb{C}'_f} \beta_h$, it can be identical to $\beta_{i,k} = \alpha_k - \sum_{F_h \in \mathbb{C}'_k} \beta_h - \alpha_f + \sum_{F_h \in \mathbb{C}'_f} \beta_h$. Thereafter, the payment p_i^F is set based on the minimum value of $\beta_{i,k}$'s. Lastly, it provides the edge devices $\boldsymbol{p}^M$ and $\boldsymbol{p}^F$ servers payment profile.

4.2.3 Offloading Model Analysis

We provide the theoretical analysis of mISO double-auction mechanism, as depicted in Alg. 1 and 2. First and foremost, we prove its truthfulness in Theorem 4.

Theorem 4. *The designed mISO multi-stage auction mechanism is truthful in nature.*

Proof. In order to the truthfulness of mISO, it needs to satisfy two conditions – (i) monotonicity and (ii) critical payment. They are discussed below:

- **Monotonicity**: The feasible cover property of Alg. 1 is autonomous to edge devices' and servers' bids, and winners can be determined considering a decreasing order of $\alpha_j - \sum i : F_i \in \mathbb{C}_j \beta_i$. Hence, if an edge device M_j wins with bidding α_j, then it can also win the auction with bidding any $\alpha_j^{'} > \alpha_j$. Furthermore, if an edge server F_i wins by bidding β_j, then the auction mechanism will be won by the edge server; besides, if the edge server bid considers any value $\beta_j^{'} > \beta_j$.
- **Critical payment**: It rewards every winning edge device and server the infimum and supremum of its bid, which will be declared as a winner in Alg. 2.

As discussed in [46], these two conditions construct the auction mechanism truthful, where each edge device M_j maximizes its utility values by bidding R_j, and each edge server F_i maximizes its utility by bidding $\mathcal{S}_i$. Hence, the mISO auction mechanism is truthful, and the proof concludes. □

Theorem 5. *The designed mISO multi-stage auction mechanism is individually rational.*

Proof. From Defs. 3 and 4, the participants who do not win in the mISO double-auction mechanism, receive zero utility values. In Theorem 4, every winning edge device M_j bids R_j, and every edge server F_i bids $\mathcal{S}_i$ to the designed platform. Furthermore, in order to win the auction mechanism, they need to pay the infimum and supremum of their bid, respectively. Therefore, this procedure assured that all the edge devices and servers attained positive utility values. Hence, the proposed mISO auction method is individually rational. □

Theorem 6. *The complexity of mISO multi-stage double-auction mechanism is $O(K^3N + K^2N^2)$.*

Proof. The complexity of Alg. 1 is $O(N)$, where it considers a greedy approach to identify the feasible covers set in line 2. Then, it identify the sets $\mathbb{C}_1, \mathbb{C}_2, \cdots, \mathbb{C}_M$, which aborted after KN times in line 3–4. Thereafter, in the worst case, the main step is aborted after K times. In order to find the edge devices' maximal social welfare, it needs $O(K)$ time, as shown in line 6. For updating the existing set $\mathbb{C}_1, \mathbb{C}_2, \cdots, \mathbb{C}_M$, it considers $O(KN)$ iterations. Hence, the complexity of the main step is $O(KN)$, and the total complexity is $O(K^2N)$. Following Alg. 1, the mISO auction mechanism runs the pricing algorithm in Alg. 2. Here, the pricing estimation step for edge devices is aborted after K times, as described in lines 1–8. The complexity of each step of the loop is influenced by running Alg. 1. Hence, the pricing estimation step for edge devices requires $O(K^3N)$ time in Alg. 2 described in lines 1–8. By using the same procedure, we also estimate the edge servers pricing in Alg. 2, which requires $O(K^2N^2)$. Therefore, the total complexity of Alg. 2 is $O(K^3N + K^2N^2)$. □

Finally, we introduce the following theorem to prove that our proposed double-auction-based mISO approach is guaranteed to have non-negative social welfare.

Theorem 7. *The mISO multi-stage auction mechanism ensures positive optimal social welfare.*

Proof. An edge device M_j and server in $\mathbb{C}_j$ are chosen as winning participants when the designating marginal social welfare $\alpha_j - \sum i : F_i \in \mathbb{C}_j \beta_i$ can be considered to be positive, as discussed in Alg. 1. Therefore, the mISO auction mechanism assures positive social welfare when the comprehensive social welfare is the summation of all aforementioned social welfare for each selected new winner as discussed in Alg. 1. Thus, the proof concludes. □

5 Experimental Results

We discuss the experiential setting of the proposed mISO framework and also benchmarks for the comparison.

5.1 Experimental Setup

We use the following settings to evaluate our proposed mISO framework, as shown in Table 4. We consider a network setting composed of 50 devices, they are randomly scattered in a region of 3 Km $\times$ 3 Km [47]–[49]. Each edge device is placed nearer to a base station, and the ratio of base stations spans over 20 to 100 meters in most cases. The edge devices act as sellers, and each seller is connected to buyers through base stations in that area. Therefore, the offloading channel is available to all the sellers nearby. In our simulation settings, buyers (i.e., edge servers) are randomly distributed in the edge computing platform. We consider around 100 edge servers. We consider in our auction mechanism that the buyers' bids are randomly distributed. If the offloading channel is not available to a seller, then the bid is considered to be 0. As the offloading channel can be reused; therefore, sellers can sell their offloading channel to more than one buyer; hence the bid can be set to higher for sellers. The compute capacity of the edge servers is 120 GHz, and the compute capacity of edge devices is 0.9 GHz. The edge servers are connected through the back-haul network with a delay coefficient 0.0001 sec/KB [50]. The microservice offloading time is set to be in the range of $15 - 20$ ms. The microservice compute traffic size is set to $500 - 900$ KB. The latency requirements of microservices are set to be within $0.8 - 1.2$ seconds.

Table 3: Variation in Experimental Contexts.

Context No.	K	N	R_j	$\mathcal{S}_i$	Θ_{ij}	$\lvert\gamma_i'\rvert$
I	50	$[50-100]$	$[5-15]$	$[5-10]$	$[0,1]$	$[10-15]$
II	$[50-100]$	$[100-150]$	$[5-15]$	$[5-10]$	$[0,1]$	$[10-15]$

Table 4: Experimental Parameters.

Parameter	Value
Base station capability (i.e., bandwidth)	25 MHz
Cycles of microservice tasks	1,500 Megacycles
Microservice deadline (i.e., completion time)	[3500, 5500] ms
Microservice resource demand in terms of bandwidth	[20, 30] MHz
Transmitter energy of edge device	95 mWatts
Compute capacity of edge users equipped with IoT devices	0.65 GHz
Compute capacity of edge servers	120 GHz
Traffic arrival rate of microservices	[0, 15] unit/sec
Compute size of traffic offloaded by microservices	120 Mbits
Microservice arrival rate following Poisson process (size = 1.5 Mbit)	[0, 15],

We take an average of 200 runs for all our simulation results. We consider the following performance metrics for our simulation results:

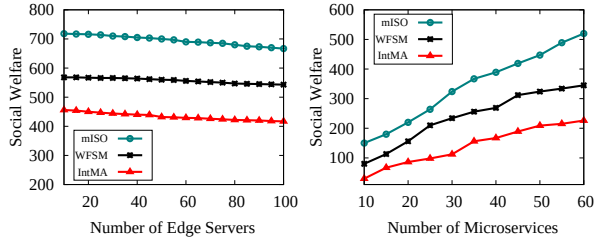
- Social welfare is determined through the incentive microservices get during offloading process.
- Buyer profit level: percentage of microservices that can get fair resources during offloading process.
- Average demand response during offloading.
- Offloading decision of microservices.

The uppermost two performance metrics show the performance of our incentive mechanism. The latter two metrics show the performance of our offloading process.

Simulator. We use EdgeCloudSim [51] to simulate the environments related to the edge computing platform and design a discrete-time simulator on top of it to evaluate the performance of mISO through a series of simulations. It simulates all the microservice events in mISO, including microservice arrival, microservice completion, microservice enqueuing, microservice dequeuing, and microservice preemption. It tracks microservice execution times.

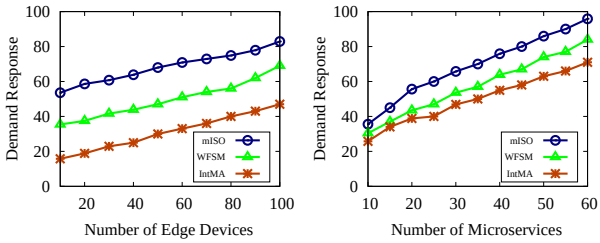
The simulator runs the algorithms at a fixed time interlude to minimize the preemption cost/overhead. The time interlude is a configurable value, which can be configured based on the performance requirements. It produces the identification labels and placement scenarios and schedules microservices according to the placement scenarios. The identification labels are generated based on the microservice identification model in Section 3.3. The mISO simulator simulates the algorithms following an executor running multiple iterations to get the average microservice execution delay and optimized results. Otherwise (without optimizations), the scheduler uses a default profile. We follow the general practice to set the number of tasks a microservice needs to be a power of two. When the identification labels and placement scenarios are created, the scheduler terminates old tasks and starts new tasks according to the scenario. The simulator tracks the demand information of each microservice, utilization, profile level, and offloading decision metric. Additionally, the simulator also tracks each microservice's enqueuing, dequeuing, and completion time.

Benchmarks We use two benchmarks - WFSM [34] proposed by Bao *et al.* and IntMA [33] proposed by Joseph *et al.*.



(a) Social Welfare (I) (b) Social Welfare (II)

Figure 3: Analysis of social welfare.



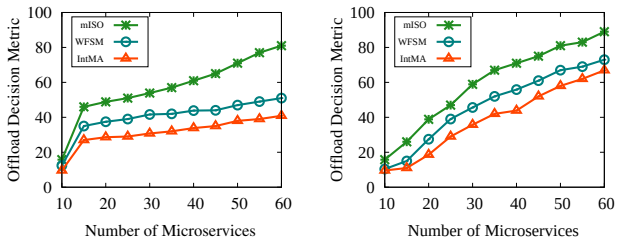
(a) Demand Response (I) (b) Demand Response (II)

Figure 4: Analysis of demand response.

WFSM [34] proposed a workflow scheduling algorithm for microservice-oriented applications in the cloud platforms. They formulated a prediction mechanism for microservices and modeled it mathematically. In this work, they basically tried to minimize the end-to-end delay while considering a predefined budget and proposed a heuristic algorithm. IntMA [33] proposed a dynamic interaction-aware resource allocation scheme for microservices. They deployed the microservice modules in the cloud platform and modeled the problem as a binary quadratic programming problem. The proposed IntMA framework, which is a novel and robust heuristic approach, considered an interaction-aware resource allocation scheme using an interaction graph. This mechanism improves the response time and throughput of microservices. We model and design the incentive mechanism for both the mechanism IntMA and WFSM while considering their respective assumptions and framework.

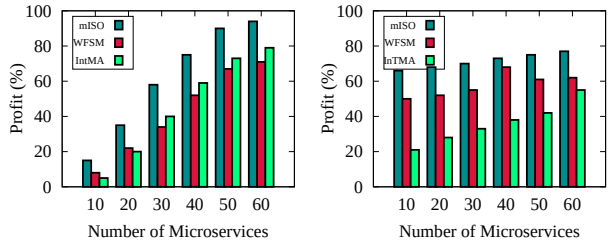
5.2 Numerical Result Analysis

Social Welfare. We compare the social welfare of the mISO incentive offloading mechanism with existing methods WFSM and IntMA. The results illustrate that the mISO incentive offloading mechanism provides better social welfare than WFSM and IntMA with contexts I and II. The social welfare of mISO is shown in Figure 3(a) with context I. As illustrated in Figure 3(a), we notice that social welfare decreases if we increase the quantity of edge servers. If the quantity of servers is scaled out in the edge platform, then the quantity of microservices executed by the server



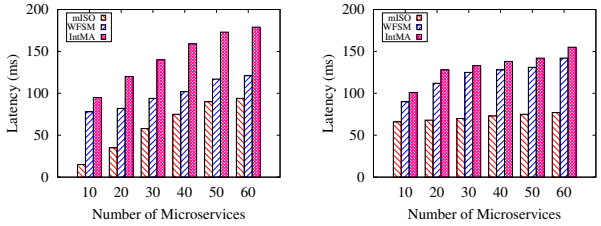
(a) Offloading Decision (I) (b) Offloading Decision (II)

Figure 5: Analysis of offloading decision.



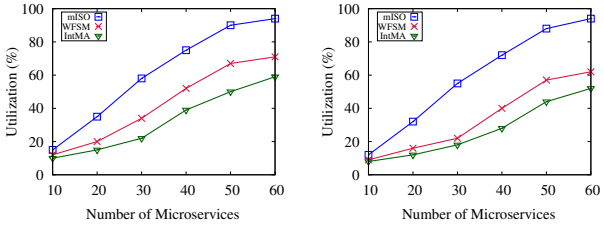
(a) Profit (I) (b) Profit (II)

Figure 6: Analysis of profit level.



(a) Latency (I) (b) Latency (II)

Figure 7: Analysis of latency.



(a) Utilization (I) (b) Utilization (II)

Figure 8: Analysis of resource utilization.

decreases. It automatically decreases the social welfare of microservice-enabled edge platforms. mISO efficiently offloads and executes microservices to servers from edge devices, which improves overall social welfare. mISO performs better than existing methods WFSM and IntMA with an improvement of 15–23 %. Figure 3(b) depicts the social welfare of mISO with context II. We can notice that social-welfare increases with the extension in the quantity of microservices. As the quantity of microservices increases, edge devices get hold of executing more quality microservices, which automatically improves the social welfare of the edge platform. It is relatively better than WFSM and IntMA. The social welfare of mISO is better than other approaches with an improvement of 13–19 %.

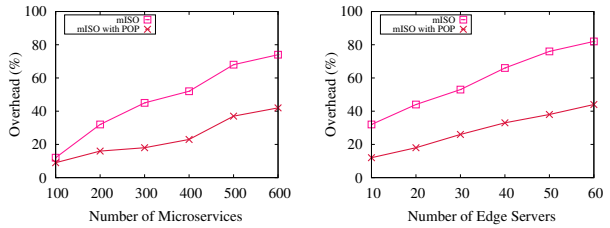
Demand Response. Figure 4(a) depicts the demand response of edge devices. It expands as the heterogeneous IoT applications increase with the context I. With the increase in the quantity of edge devices, the quantity of microservices spawned up in the edge servers inevitably increases the demand response of edge devices. mISO achieves better performance than existing methods WFSM and IntMA. The demand response of edge devices of mISO is better than other methods with an improvement of 18–25 %. Further, Figure 4(b) illustrates the demand response of edge devices with the extension in the quantity of microservices with context II. As the quantity of microservices extends in the platform, then the resource required to execute the microservices also expands, it provides a steep rise in the demand response. mISO shows better performance than other methods, WFSM and IntMA, with an improvement of 9–11 %.

Offloading Decision. Figure 5(a) signifies the offloading decision of mISO with the context I. We can notice from the figure that offloading decision metric improves with expanding the quantity of microservices. As the quantity of microservices expands, the microservice flow rate also expands. Therefore, the offloading decision metric also expands, as it is a factor of data traffic function. mISO effectively improves the offloading performance of microservices to servers. mISO basically outruns the other methods, WFSM and IntMA, with an improvement of 8–13 %. Further, we also check out the offloading decision with context II in Figure 5(b). It also outruns the other methods, WFSM and IntMA, with an improvement of 7–9 %.

Profit Level. Figure 6(a) signifies the profit level of mISO with the context I. We discover that the profit level of edge devices improves with expansion in the quantity of microservices. As the quantity of microservices expands, then the mISO can process more quantity of microservices to servers. The profit of mISO is superior to other methods with an improvement of 13–18 %. On the other hand, Figure 6(b) depicts the profit level of mISO with context II. mISO outruns the existing methods WFSM and IntMA. The profit level of mISO is better than other methods with an improvement of 16–26 %.

Latency. Figure 7(a) signifies the average latency of mISO with the context I. We discover that the average latency of edge devices improves with expansion in the quantity of microservices. As the quantity of microservices expands, then the mISO can process more quantity of microservices to servers. The average latency of mISO is lesser than other methods with an improvement of 20–22 %. On the other hand, Figure 7(b) depicts the average latency of mISO with context II. mISO outruns the existing methods WFSM and IntMA. The average latency of mISO is better than other methods with an improvement of 18–21 %.

Utilization. Figure 8(a) signifies the resource utilization of mISO with the context I. We discover that the resource utilization of edge devices improves with the expansion in the quantity of microservices. As the quantity of microservices expands, then the mISO can process more quantity of



(a) Scalability (b) Scalability

Figure 9: Analysis of Overhead.

microservices to servers. The resource utilization of mISO is superior to other methods with an improvement of 21-24 %. On the other hand, Figure 8(b) depicts the average latency of mISO with context II. mISO outruns the existing methods WFSM and IntMA. The utilization of mISO is better than other methods with an improvement of 25-30 %.

Overhead. We can observe from the Theorem 6 that the complexity of mISO is $O(K^3N+K^2N^2)$. For a large number of microservices and edge servers, the system overhead of mISO would be very high. Hence, it is important to minimize the system overhead for a large number of microservices and edge servers. Figure 9(a) shows that the overhead of mISO with Partitioned Optimization Problems (POP) [52] is comparatively very low than standalone mISO with expansion in the quantity of microservices and 20 edge servers. By adopting POP, we can randomly split the problem into smaller problems (with a subset of the clients and resources in the server) and combine the resulting subproblem into a global offloading for all clients. POP is easy to implement and provides better runtime. Similarly, mISO with POP can achieve 32 % less overhead compared to standalone mISO with increasing edge servers and 300 microservices.

5.3 Discussion

We enlist a few main takeaways from the mISO mechanism.

- mISO provides 18–21 % and 25–30 % improvements in terms of average latency and resource utilization compared to existing works. mISO's benefits are due to efficient microservice identification, resilient demand estimation, and optimal microservice offloading algorithms.
- mISO mechanism holds truthfulness, rationality, and low computational complexity while guaranteeing positive social welfare.
- mISO is robust to various configuration parameters and experimental settings.

We enlist a few limitations of the mISO mechanism.

- mISO cannot handle any concerns regarding the failures. Therefore, if a single task fails within a microservice, then the entire microservice can fail. To tackle such situations, we need a resilient load-balancing scheme or fault-tolerant mechanism.
- mISO does not consider any concerns related to designing an energy-efficient incentive mechanism. This may lead to some problems if there are shortage of resources and edge devices keep on trying to offload the microservices; in such cases, the edge devices' batteries may go down. So in the future, we need an energy-efficient incentive mechanism.

6 Conclusion

We present a novel incentive mechanism for MEC-enabled IoT networks to provide maximum social welfare to edge IoT devices. mISO proposes a demand-agnostic scheme, which efficiently estimates the demand specifications of edge devices to maximize the profit of the network. Further, we come up with an incentive microservice offloading scheme to optimize the social welfare of both edge devices and servers. Hence, the proposed scheme–mISO efficiently offloads the computational microservices to edge servers to increase the system utilization. Rigorous and extensive simulation results show that mISO achieves better social welfare than the existing methods WFSM and IntMA. Building a real system implementation for optimizing the microservice allocation problem in complex and dynamic MEC systems is a promising topic for future research. Another interesting topic is an adaptive microservice placement for MEC-based mobile health systems.

References

[1] Q.-V. Pham, F. Fang, V. N. Ha, M. J. Piran, M. Le, L. B. Le, W.-J. Hwang, and Z. Ding, "A survey of multi-access edge computing in 5G and beyond: Fundamentals, technology integration, and state-of-the-art," *IEEE Access*, vol. 8, pp. 116 974–117 017, 2020.

[2] A. Samanta and Y. Li, "Latency-Oblivious Incentive Service Offloading in Mobile Edge Computing," in *Proc. of IEEE/ACM SEC*. IEEE, 2018, pp. 351–353.

[3] D. Xu, A. Samanta, Y. Li, M. Ahmed, J. Li, and P. Hui, "Network coding for data delivery in caching at edge: Concept, model, and algorithms," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 10, pp. 10 066–10 080, 2019.

[4] A. Brogi, S. Forti, and A. Ibrahim, "How to best deploy your fog applications, probably," in *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*, 2017, pp. 105–114.

[5] F. Alkhabbas, I. Murturi, R. Spalazzese, P. Davidsson, and S. Dustdar, "A goal-driven approach for deploying self-adaptive iot systems," in *2020 IEEE International Conference on Software Architecture (ICSA)*, 2020, pp. 146–156.

[6] Z. Yang, M. Chen, K. Wong, H. V. Poor, and S. Cui, "Federated learning for 6g: Applications, challenges, and opportunities," *CoRR*, vol. abs/2101.01338, 2021. [Online]. Available: https://arxiv.org/abs/2101.01338

[7] Z. Yang, C. Pan, K. Wang, and M. Shikh-Bahaei, "Energy efficient resource allocation in uav-enabled mobile edge computing networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 9, pp. 4576–4589, 2019.

[8] A. Samanta and Y. Li, "Time-to-think: Optimal economic considerations in mobile edge computing," in *Proc. of IEEE INFOCOM Workshops*, 2018, pp. 1–2.

[9] J. Cho, K. Sundaresan, R. Mahindra, J. Van der Merwe, and S. Rangarajan, "Acacia – Context-aware Edge Computing for Continuous Interactive Applications over Mobile Networks," in *Proc. of MobiCom*, 2016, pp. 505–506.

[10] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2016.

[11] M. Satyanarayanan, "The Emergence of Edge Computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[12] A. Samanta and J. Tang, "Dyme: Dynamic microservice scheduling in edge computing enabled IoT," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[13] A. Samanta, T. G. Nguyen, T. Ha, and S. Mumtaz, "Distributed resource distribution and offloading for resource-agnostic microservices in industrial iot," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 1, pp. 1184–1195, 2022.

[14] N. Cruz Coulson, S. Sotiriadis, and N. Bessis, "Adaptive microservice scaling for elastic applications," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4195–4202, 2020.

[15] A. Samanta, F. Esposito, and T. G. Nguyen, "Asap: Adaptive and scalable microservice provisioning for edge-iot networks," in *2023 18th Wireless On-Demand Network Systems and Services Conference (WONS)*, 2023, pp. 80–87.

[16] M. Jia, W. Liang, Z. Xu, and M. Huang, "Cloudlet Load Balancing in Wireless Metropolitan Area Networks," in *Proc. of IEEE INFOCOM*, 2016, pp. 1–9.

[17] Y. Xiao, M. Noreikis, and A. Ylä-Jaäiski, "QoS-oriented Capacity Planning for Edge Computing," in *Proc. of IEEE ICC*, 2017, pp. 1–6.

[18] A. Samanta, Z. Chang, and Z. Han, "Latency-oblivious distributed task scheduling for mobile edge computing," in *Proc. of IEEE GLOBECOM*, 2018, pp. 1–7.

[19] M. Gao, R. Shen, J. Li, S. Yan, Y. Li, J. Shi, Z. Han, and L. Zhuo, "Computation offloading with instantaneous load billing for mobile edge computing," *IEEE Transactions on Services Computing*, pp. 1–1, 2020.

[20] D. T. Nguyen, L. B. Le, and V. Bhargava, "Price-based resource allocation for edge computing: A market equilibrium approach," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2018.

[21] Y. Zhang, X. Lan, J. Ren, and L. Cai, "Efficient computing resource sharing for mobile edge-cloud computing networks," *IEEE/ACM Transactions on Networking*, vol. 28, no. 3, pp. 1227–1240, 2020.

[22] X. Ma, S. Wang, S. Zhang, P. Yang, C. Lin, and X. S. Shen, "Cost-efficient resource provisioning for dynamic requests in cloud assisted mobile edge computing," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2019.

[23] Q. Wang, S. Guo, J. Liu, C. Pan, and L. Yang, "Profit maximization incentive mechanism for resource providers in mobile edge computing," *IEEE Transactions on Services Computing*, pp. 1–1, 2019.

[24] J. Liu, S. Guo, K. Liu, and L. Feng, "Resource provision and allocation based on microeconomic theory in mobile edge computing," *IEEE Transactions on Services Computing*, pp. 1–1, 2020.

[25] X. Hou, Z. Ren, J. Wang, W. Cheng, Y. Ren, K. Chen, and H. Zhang, "Reliable computation offloading for edge computing-enabled software-defined IoV," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[26] Q.-V. Pham, T. Leanh, N. H. Tran, B. J. Park, and C. S. Hong, "Decentralized computation offloading and resource allocation for mobile-edge computing: A matching game approach," *IEEE Access*, vol. 6, pp. 75 868–75 885, 2018.

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

[28] P. Lai, Q. He, J. Grundy, F. Chen, M. Abdelrazek, J. G. Hosking, and Y. Yang, "Cost-effective app user allocation in an edge computing environment," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2020.

[29] L. Chen, C. Shen, P. Zhou, and J. Xu, "Collaborative service placement for edge computing in dense small cell networks," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2019.

[30] Y. Li, X. Wang, X. Gan, H. Jin, L. Fu, and X. Wang, "Learning-aided computation offloading for trusted collaborative mobile edge computing," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2019.

[31] A. Samanta and Y. Li, "DeServE: Delay-agnostic Service Offloading in Mobile Edge Clouds: Poster," in *Proc. of ACM/IEEE SEC*, 2017, pp. 24:1–24:2.

[32] A. Samanta, L. Jiao, M. Mühlhäuser, and L. Wang, "Incentivizing microservices for online resource sharing in edge clouds," in *IEEE ICDCS*, 2019, pp. 1–9.

[33] C. T. Joseph and K. Chandrasekaran, "IntMA: Dynamic interaction-aware resource allocation for containerized microservices in cloud environments," *Journal of Systems Architecture*, vol. 111, p. 101785, 2020.

[34] L. Bao, C. Wu, X. Bu, N. Ren, and M. Shen, "Performance modeling and workflow scheduling of microservice-based applications in clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 9, pp. 2114–2129, 2019.

[35] S. Wang, Y. Guo, N. Zhang, P. Yang, A. Zhou, and X. S. Shen, "Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach," *IEEE Transactions on Mobile Computing*, vol. 20, no. 3, pp. 939–951, 2019.

[36] M. Abdullah, W. Iqbal, J. L. Berral, J. Polo, and D. Carrera, "Burst-aware predictive autoscaling for containerized microservices," *IEEE Transactions on Services Computing*, vol. 15, no. 3, pp. 1448–1460, 2022.

[37] M. Abdullah, W. Iqbal, A. Mahmood, F. Bukhari, and A. Erradi, "Predictive autoscaling of microservices hosted in fog microdata center," *IEEE Systems Journal*, pp. 1–12, 2020.

[38] G. Yu, P. Chen, and Z. Zheng, "Microscaler: Cost-effective scaling for microservice applications in the cloud with an online learning approach," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2020.

[39] A. Samanta, Y. Li, and F. Esposito, "Battle of microservices: Towards latency-optimal heuristic scheduling for edge computing," in *IEEE Conference on Network Softwarization (NetSoft)*, 2019, pp. 223–227.

[40] Z. Wen, T. Lin, R. Yang, S. Ji, R. Ranjan, A. Romanovsky, C. Lin, and J. Xu, "GA-Par: Dependable microservice orchestration framework for geo-distributed clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 1, pp. 129–143, 2020.

[41] R. P. McAfee, "A Dominant Strategy Double Auction," *Journal of Economic Theory*, vol. 56, no. 2, pp. 434–450, 1992.

[42] H. Jin, L. Su, and K. Nahrstedt, "CENTURION: Incentivizing Multi-Requester Mobile Crowd Sensing," in *Proc. of IEEE INFOCOM*, 2017.

[43] X. Feng, Y. Chen, J. Zhang, Q. Zhang, and B. Li, "TAHES: Truthful double Auction for Heterogeneous Spectrums," in *Proc. of IEEE INFOCOM*, 2012, pp. 3076–3080.

[44] Z. Zheng and N. B. Shroff, "Online Multi-Resource Allocation for Deadline Sensitive Jobs with Partial Values in the Cloud," in *Proc. of IEEE INFOCOM*. IEEE, 2016, pp. 1–9.

[45] D. P. Bertsekas, *Nonlinear Programming*. Athena scientific Belmont, 1999.

[46] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic Game Theory*. Cambridge University Press Cambridge, vol. 1.

[47] A. Samanta and Z. Chang, "Adaptive Service Offloading for Revenue Maximization in Mobile Edge Computing with Delay-Constraint," *IEEE Internet of Things Journal*, pp. 1–1, 2019.

[48] A. Samanta and Y. Li, "Deserve: delay-agnostic service offloading in mobile edge clouds: Poster," in *ACM/IEEE SEC*, 2017, p. 24.

[49] A. Samanta, F. Esposito, and T. G. Nguyen, "Fault-tolerant mechanism for edge-based iot networks with demand uncertainty," *IEEE Internet of Things Journal*, vol. 8, no. 23, pp. 16 963–16 971, 2021.

[50] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Energy-Efficient Offloading for Mobile Edge Computing in 5G Heterogeneous Networks," *IEEE Access*, vol. 4, pp. 5896–5907, 2016.

[51] C. Sonmez, A. Ozgovde, and C. Ersoy, "Edgecloudsim: An environment for performance evaluation of edge computing systems," *Wiley Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 11, p. e3493, 2018.

[52] D. Narayanan, F. Kazhamiaka, F. Abuzaid, P. Kraft, A. Agrawal, S. Kandula, S. Boyd, and M. Zaharia, "Solving large-scale granular resource allocation problems efficiently with pop," in *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*, 2021, pp. 521–537.