Fault-Tolerant Mechanism for Edge-Based IoT Networks with Demand Uncertainty

Amit Samanta, Flavio Esposito, *Member, IEEE* and Tri Gia Nguyen, *Member, IEEE*

***Abstract*—Due to ubiquitous increase of mobile services and powerful Internet-of-Things (IoT) devices, the interest for Mobile Edge Computing (MEC) solutions has grown both in industry and academia. One of the fundamental mechanisms of MEC is offloading, i.e., delegation of a computation from the user to a server (set) placed near to the edge. Edge servers may have poor incentives to run a delegated service, for example for the temporary limited resources. In this paper, we dissect the incentive mechanisms within a MEC ecosystem with the aim of ensuring a fault-tolerant edge service under unreliable scenarios. In particular, we design an auction mechanism to model the interaction between the MEC players, and model the edge users' probability of successful offloading, assuming that the cost of executing each offloading request is private. Scrutinizing the demand uncertainty of edge users, the main motive of our auction method is to optimize the offloading cost to engage more edge users in this process, while imposing probabilistic guarantees of offloading service execution. Our offloading cost minimization problem is considered to be a NP-hard. For the solution, we use a heuristic methodology to get the optimal approximation ratio and provide economical fairness. We provide exhaustive simulation results to show the excellent performance of our scheme.**

***Index Terms*—Fault-tolerant, Internet of Things, IoT, Edge computing, Social-welfare.**

I. Introduction

Mobile Edge Computing (MEC) has evolved from cloud computing and is today a vital paradigm for several services and applications. The fundamental objective of MEC [1]–[4] is to deploy small computational capabilities at the edge of the network, where IoT devices run several computational-intensive services to support heterogeneous applications [5]. While such paradigm brought many advantages, several challenges still hinder the edge cloud ecosystem: (i) the highly dynamic and uncertain network condition, causing in turn, high overhead due to intermittent connectivity and node failures; (ii) the often constrained resources and capacity of edge devices, (iii) the variability in resource execution and unknown performance utilization.

Because of such limitations, edge infrastructure processes may fail to execute their assigned computation. These uncertainties lead to Service Level Agreement violations for edge users and revenue loss for the service and infrastructure provider. Hence, it is important to identify such resource uncertainties and prevent suboptimalities. In this paper, we examine service offloading under demand uncertainty. We assume that each IoT device is permitted to offload a computational service with a given probability, and design a strategy-proof mechanism whose aim is to optimize the total execution cost of the services executed at the edge, while providing higher probability of success during the offloading process. Designing such mechanism entails solving several challenges. The classical offloading model assumes that IoT devices choose which server to use to outsource their excessive computations. To make such decisions, all players need to estimate the amount of resources available. As in any other phenomena modeled as a game, the objective of selfish and rational edge users is to optimize the respective net utility. IoT devices may have incentive to report false demand response in order to increase their net utilities or lower their cost. Our cost captures infrastructure computational costs or resource demand, including monetary cost. Computing some of these costs, e.g., the service executing cost, is however, a challenge. We show that minimizing the offloading cost of edge services, while not examining the tactical behavior of edge users is NP-hard. This in turn, motivates us to explore for an efficient and optimal algorithm. Existing approximation methods, e.g., [6] do not provide any assured and robust economic properties. Therefore, to continuously attain such higher approximation ratio and robust economic performances in mechanism design while considering the demand uncertainty is the challenge that we explore in this paper. In particular, our contributions are summarized as follows:

(1) We design a strategy-optimal and economical mechanism design for edge users in MEC platform while considering demand uncertainty in service execution. Our aim is to optimize the offloading cost providing guaranteed service execution with higher probability. (2) We propose DECADE*, a mechanism that estimates the demand response of edge users to minimize the services delay using a demand service approximation profile. (3) We design an optimal payment (i.e. utility) scheme for IoT devices that functions with demand unpredictability. We mathematically show that such payment method is computationally effective, and it assures optimal strategy. (4) Finally, we conduct an extensive simulation campaign to evaluate our fault-tolerant mechanism based on two types of services, delay-tolerant and delay-sensitive. Our simulation results depicts that our mechanism outruns the existing solutions. Our mechanism also provide probabilistic

Amit Samanta is with the School of Computing, University of Utah, USA. (amit.samanta049@gmail.com)

Flavio Esposito is with the Department of Computer Science, Saint Louis University, USA. (flavio.esposito@slu.edu)

Tri Gia Nguyen (corresponding author) is with the Department of Information Technology, FPT University, Da Nang 50509, Vietnam. (tri@ieee.org)

*DECADE: Fault-Tolerant Mechanism **D**esign for **E**dge **C**omputing IoT Networks with **DE**mand uncertainty.

bounds on the execution of the offloaded task even with challenged network scenarios.

The rest of the paper is organized as follows. In Section II, we provide the related work. We present a model for fault-tolerant mechanism design in Section III and presents a demand approximation model to estimate the demand uncertainty of edge computing users. In Section IV, we formulate an auction to capture the interaction between the edge infrastructure and the set of edge users. Our results are presented in Section V and in Section VI we conclude the paper.

II. Related Work

A. Resource optimization and offloading for MEC

In [7], Samanta *et al.* propose a service offloading approach considering the revenue maximization problem in MEC platform. The aims of the work are to efficiently perform service offloading proposing a utility maximization approach. A distributed multi-channel computation offloading algorithm is proposed by Chu *et al.* [8]. The authors first formulated computation offloading problems consider the game-theoretic analysis and Nash equilibrium possessing. The authors then proposed a game-theoretic algorithm for multi-channel computation offloading to get a Nash equilibrium point. In [9], Wang *et al.* focuses on solving the resource allocation scheme by developing an edge node resource management framework for fog computing. Hu *et al.* [10] proposed a heterogeneous offloading mechanism without knowing the complete information. In [11], Xia *et al.* focuses on the task scheduling problem at the edge that considers Quality of Experience (QoE). The authors formulate a mixed-integer non-linear programming problem as a model of the task scheduling problem. The authors then propose a scheduling approach to optimize task allocation based on QoE. These solutions mainly focuses on the resource provisioning and sharing among multiple edge applications. However, they did not consider the demand of available and required resources of edge cloud platform and applications. Hence, in this paper, we propose a demand approximation method by which accurately identify available and required resources of edge cloud platform.

B. IoT Networks

Work on mechanism design for IoT networks also exists [12]–[17]. For example, Xiong *et al.* [12] aim at reducing communication overhead and computation cost, while the authentication protocol ensures security and privacy. In [13], Liao *et al.* focus on addressing joint optimization problem in MEC for IoT. The authors formulate a dynamic pricing approach based on Markov decision process. The authors then address the decision-making problem by using a Q-learning algorithm. In [14], Fantacci and Picano focus on minimizing the average system response time and the number of dropping requests in edge-cloud computing for IoT. The authors first formulate a matching game based on the application requested by the edge-computing servers and the IIoT devices, then propose a virtual machine placement solution to solve the matching game minimizing the average system response time. A formalization approach of the cloud-edge allocation problem for IIoT is presented by Casola *et al.* [15]; the authors formalize the cloud-edge allocation problem considering security, cost, and performance aspects. Similarly, Zhou [18] proposed a cost-effective edge resource provisioning mechanism for IoT applications. All these solutions mainly focuses on optimizing the resources and delay for heterogeneous edge applications. However, there is a need of a system that aims at maximizing profits of both edge platforms and IoT devices. Along with it, none of the existing work considered a fault-tolerant mechanism that deals with resource uncertainly of edge platform. Our design fills this gap adding value to edge ecosystem.

Synthesis. Most of the existing literature in edge computing focuses on the computational offloading techniques and minimization of service delay in MEC [7]–[11], [19], [20]. None of the existing literature considers the resource-agnostic property of the IoT devices and moreover have not proposed any optimal and flexible resource distribution scheme for IoT devices to optimize the execution cost and fault-tolerance. Though in the existing literature have many resource distribution techniques, but mainly they are limited to general wireless and cellular networks. However, these are techniques not deemed to fit for edge computing.

III. System Model and Demand Approximation

System Model. As in prior work [21], [22] we model an edge network considering several "offloaders" or edge users, a centralized orchestrator (or broker), and several edge servers. Using such model, we design a fault-tolerant mechanism to provide the optimal latency and the service completion time (SCT) under demand uncertainty of IoT devices. The centralized platform collects a set of edge service offloading requests $\mathbb{S} = \{1, 2, \cdots, S\}$ from different edge users, and aims at offloading such requests to the optimal edge server set. Each edge service $i \in \mathbb{S}$ is associated with a probability $\mathbb{P}_i$ of being chosen by the centralized orchestrator. Each edge user executes its services with probability $\mathbb{P}_i$. The edge network also consists of a set of edge users $\mathbb{E} = \{1, 2, \cdots, E\}$ equipped with modern IoT devices. Each edge user $j \in \mathbb{E}$, has a set of services $\mathcal{H}_i \in \mathbb{S}$ to offload and execute successfully; each service has an application type and other features and constraints (i.e., execution time, requested quality-of-service and priority level). The edge user $j \in \mathbb{E}$ has a probability of successful offloading λ_j^i for each service $i \in \mathbb{S}_j$, and a total execution cost $\mathcal{C}$ to execute all the services $\mathbb{S}_j$. We can measure the Probability of Execution (PoE) under circumstances of different situations. As an example, in service offloading, the PoE of service of an edge user is defined as the probability of offloading and executing the services opportunistically.

Edge users try to execute all their services despite a probability of failure to complete some of them (e.g., due to under demand uncertainty or link-failure). We assume that each edge user can approximate such values using their previous data records. We model this information to be, however private to each user. The execution cost is derived from an edge user executing their services and depends on factors such as offloading cost, migration cost, and data collection cost. To design our mechanism, we model the profile Υ_j of an edge

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3075681, IEEE Internet of Things Journal

user $j \in \mathbb{E}$ with three tuples: service set, execution cost, and PoE's, that are:

$$\Upsilon_j = (\mathbb{S}_j, \mathcal{C}_j, \{\lambda_j^i | i \in \mathbb{S}_j\}).$$

We consider $-j$ in subscript to identify all edge users excluding j, e.g., Υ_{-j} shows the types of all users excluding j, and $\Upsilon = (\Upsilon_j; \Upsilon_{-j})$ shows the aggregated trace of edge users. For each user $j \in \mathbb{E}$, the net utility $\mathcal{U}_j$ is defined as the reward $\mathcal{H}_j$ received, deducting the execution cost of existing services, $\mathcal{U}_j = \mathcal{H}_j - \mathcal{C}_j$. We design a scenario where each edge user maximizes their net utility and may misreport its demand response to gain additional benefits or merely due to a system misconfiguration.

Demand Approximation. We now describe how to approximate the demand of all the edge IoT devices simultaneously active at any given time, given the network traffic and workload observed by each edge cloud process. We assumed that different IoT devices have different demands and they intend to misguide all other users to prevent them from executing their services. This assumption is important to approximate the demands of IoT devices and to allocate the resources optimally, as edge platform are generally resource-constrained.

To estimate such offloading demand of each edge user associated with each IoT device, the orchestrator needs to sustain a global feedback trace of each edge user in a time frame $\bar{\tau} = \{\tau_1, \tau_2, \cdots, \tau_n\}$. Note how here, n illustrates the count of samples considered for the approximation mechanism. We denote this quantity as:

$$\mathcal{F} = \{\mathbb{F}_1^{\tau_1}, \mathbb{F}_2^{\tau_2}, \cdots, \mathbb{F}_n^{\tau_n}\}. \tag{1}$$

As demands are dynamic, we aim at quantifying the rate of demand change for a particular time period $\bar{\tau}$. We denote such rate as $\zeta_j = \delta\mathcal{W}(\mathbb{F}_j, \tau_n)$, where τ_n is offloaded time of edge user E_j. Mathematically:

$$\zeta_j = \delta\mathcal{W}(\mathbb{F}_j, \tau_n) = \delta\left(\frac{\mathbb{F}_j^{\tau_n} - \mathbb{F}_j^{(\tau_n - 1)}}{\mathbb{F}_j^{\tau_n}}\right), \tag{2}$$

where $\mathbb{F}_j^{\tau_n}$ and $\mathbb{F}_j^{(\tau_n-1)}$ signifies the demand specification of edge user E_j in time-slot τ_n and $\tau_n - 1$, respectively. This equation captures the demand change rate of IoT devices at $\bar{\tau}$. Further, we get an optimal demand specification, $\mathcal{G}_j(\tau+1)$ of edge user E_j in time-slot $\tau + 1$ using the following formula.

$$\mathcal{G}_j(\tau+1) = \mathcal{G}_j(\tau) + \sum_{k=1}^{n} \frac{\eta_j(\tau)\zeta_j(i) f_j^t(x')}{\zeta_j(i) + \eta_i^j(k)\zeta_j(i)}, \tag{3}$$

where $\mathcal{G}_j(\tau)$ denotes the total demand response of edge user E_j in time-slot τ and $\eta_j(\tau)$ denotes the resource efficiency constant of edge user B_j. $f_j^t(x')$ denotes the unit service flow of edge user E_j in time-slot τ. The resource efficiency constant $\eta_j(\tau)$ of edge user E_j in time-slot τ is discussed:

$$\eta_j(\tau) = \begin{cases} 1, & \varsigma_{th}(\tau) > 0; \\ 0, & \varsigma_{th}(\tau) < 0;, \end{cases} \tag{4}$$

where $\eta_{th}(\tau)$ signifies the threshold resource efficiency constant. The service flow $f_j^t(x')$ of edge user E_j in time-slot τ can be formulated based on the edge service flow and mean time length of service flow. We get,

$$f_j^t(x') = \begin{cases} \frac{\mathcal{Q}_j^t}{\bar{\tau}}, & \text{if } \eta_j(t) \geq 1; \\ \frac{\mathcal{Q}_j^t}{\bar{\tau}}(1 - \gamma_j), & \text{if } \eta_j(t) < 0, \end{cases} \tag{5}$$

where $\mathcal{Q}_j^t$ denotes the unit service flow. $\bar{\tau}$ signifies the entire time duration of demand approximation. γ_j illustrates the retribution cost of service flow. It is imposed, if the resource efficiency constant is closer to $\eta_j(\tau) = 0$. This is because with the decrease in efficiency constant, the demand of IoT device suffers performance degradation. The unit service flow from each edge user E_j in time-slot τ is represented as:

$$\mathbb{Q}_j^\tau = V_j \frac{\mathcal{P}_{tra}^j(\tau)\mathbb{O}}{\tau_{tra} + \Delta\tau}, \tag{6}$$

where V_j illustrates the unit service flow rate of edge user E_j and $\tau_{tra} + \Delta\tau$ illustrates the total execution duration of service flow. $\mathcal{P}_{tra}^j(\tau)$ illustrates the total count of unit service flow executed and $\mathbb{O}$ illustrates the flow size. The service flow of edge users can be formulated based on the traffic description and it's type. We consider three types of traffic - *high class*, *moderate class* and *low class*. Depending on traffic descriptions, the service flow of edge users is formatted as:

$$\mathcal{Q}_i^\tau = \begin{cases} \mathcal{Q}_l^\tau, & \text{if } V_j \geq 0.9; \\ \mathcal{Q}_m^\tau, & \text{if } V_j \geq 0.5; \\ \mathcal{Q}_h^\tau, & \text{if } V_j \geq 0.1; \\ 0, & \text{if } V_j = 0, \end{cases} \tag{7}$$

where $\mathcal{Q}_l^\tau$, $\mathcal{Q}_m^\tau$, and $\mathcal{Q}_h^\tau$ defines the lower class service flow, moderate class service flow, and higher class service flow. The accurate resource necessity $\nu_j^t\big(\eta_j(\tau)\zeta_j(i)(1-\mu) + \eta_j(\tau)\zeta_j(i)\sigma_j(1-\mu)\big)$ should be higher than the minimal resource necessity, ν_m of edge users. We have,

$$\sum_{j=1}^{E}\sum_{\tau=1}^{T} \nu_j^\tau \Big[\eta_j(\tau)\zeta_j(i)(1-\mu) + \eta_j(\tau)\zeta_j(i)\sigma_j(1-\mu)\Big] \geq \nu_m,$$

where ν and ν_m signifies the accurate resource necessity and minimal resource necessity of edge user E_j in time-slot τ. μ signifies the loss probability. σ signifies the service priority of edge users ($E_j \in [0, 1]$). Depending on accurate demand of edge users, the platform analyzes the workload of each edge user in order to execute their services. With consideration the demand feedback of edge users, the platform offloads the services and schedules them to a particular edge user to fulfill their accurate demand. We define the optimization problem for demand approximation of IoT devices. We have,

$$\text{Maximize } \mathcal{G}_j(t+1) = \mathcal{G}_j(t) + \sum_{k=1}^{n} \frac{\eta_j(t)\zeta_j(i) f_j^t(x')}{\zeta_j(i) + \eta_i^j(k)\zeta_j(i)}, \tag{8}$$

$$\text{Subject to} \quad \mathcal{Q}_j^t \leq \mathcal{Q}_{th}^t, \forall j \tag{9}$$

$$\gamma_j > \gamma_{th}, \forall j \tag{10}$$

$$\mathbb{F}_j \geq \mathbb{F}_{th}, \forall j \tag{11}$$

$$f_j^t(x'), \eta_j(t) \in \{0, 1\}, \forall j \tag{12}$$

Equation (8) describes the objective function for our IoT demand approximation. Equation (9) constraints the service

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3075681, IEEE Internet of Things Journal

flow rate $\mathcal{Q}_j^t$ to a value greater than the threshold service flow rate $\mathcal{Q}_{th}^t$. The penalty cost γ_j is imposed to be lower than the threshold penalty cost $\gamma_y th$ as shown in Equation (10). The required demand $\mathbb{F}_j$ to be greater than the threshold demand $\mathbb{F}_{th}$ of IoT devices as in Equation (11).

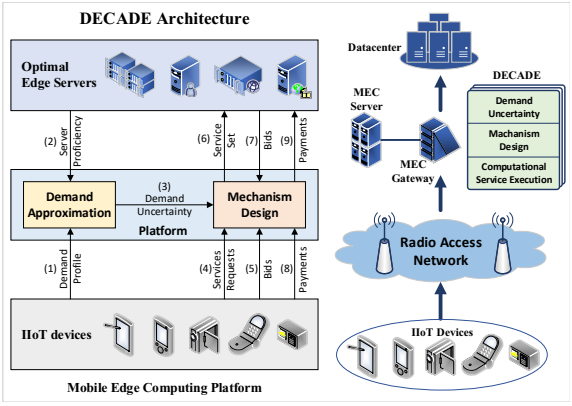


Figure 1: Architectural overview of DECADE, where the numbers define the occurrence of corresponding events.

IV. Fault-Tolerant Mechanism Design

This section introduces our DECADE system architecture (Figure 1). DECADE estimates the demand of edge services and then implements the problem defined in Equation (19) with a fully distributed auction. Each edge service is aware just of the portion of solution under its demand uncertainty and responsibility.

In our auction mechanism, each IoT device E_j provides a service S_j to the designed platform for efficient execution. Our architecture then reports the set of services $\bar{S}$ to edge servers. Each edge server K_l collects and runs such services. The edge servers also submit a bid z_i to execute the services efficiently. After receiving the bids, the incentive module clears the device/edge server allocation; the remittance c_j^E is charged from each auction winner IoT device E_j, and a payment c_l^K is submitted to each winner edge server K_l. Thereafter, the designed platform accumulates the executed services submitted by the IoT devices and collects the accumulated results. Then, the accumulated results are sent to the successful IoT devices for further processing. Here, we consider two different bidding profile for IoT devices and servers denoted as $\boldsymbol{y} = \{y_1, y_2, \cdots y_N\}$ and $\boldsymbol{z} = \{z_1, z_2, \cdots z_L\}$, respectively. Further, we also consider two payment profiles for IoT devices and servers denoted as $\boldsymbol{c}^E = \{c_1^E, c_2^E, \cdots c_N^E\}$ and $\boldsymbol{c}^K = \{c_1^K, c_2^K, \cdots c_L^K\}$.

Definition 1. *In an auction mechanism for MEC, IoT device E_j gets a value M_j if the service S_i is successfully execution, and bids to the designed platform y_j, the amount the IoT device is willing to pay for the successful execution of its service. Similarly, each server K_l is interested in execution of one subset of the services $S_i \in \bar{S}$, and bids to the designed platform z_l the edge server bidding price for executing all the services. The real execution cost for executing all the servers K_l is denoted as $\mathcal{C}_l$. Both the IoT devices' and servers' bids are private.*

Next, we define the utility functions for IoT devices and servers and the profit level.

Definition 2. *The utility function $\mathcal{U}_j^E$ for an IoT device E_j is defined as:*

$$\mathcal{U}_j^E = \begin{cases} M_j - c_j^E, & E_j \in \mathcal{L}_{\mathbb{E}}, \\ 0, & otherwise, \end{cases} \tag{13}$$

Definition 3. *The utility function $\mathcal{U}_i^K$ for an edge server K_l is defined as:*

$$\mathcal{U}_l^K = \begin{cases} c_l^K - \mathcal{C}_l, & K_l \in \mathcal{L}_{\mathbb{K}}, \\ 0, & otherwise, \end{cases} \tag{14}$$

Definition 4. *The profit level is defined as:*

$$\mathbb{P}_{le} = \sum_{j:E_j \in \mathcal{L}_{\mathbb{E}}} c_j^E - \sum_{l:K_l \in \mathcal{L}_{\mathbb{K}}} c_l^K \tag{15}$$

Based on definitions 2, 3, and 4, we formulate the social welfare of the MEC platform, as the summation of the designed platform's profit levels and all IoT devices' and servers' utility values, that is:

Definition 5. *The social welfare is defined as:*

$$\mathbb{S}_{wel} = \mathbb{P}_{le} + \sum_{j:E_j \in \mathcal{L}_{\mathbb{E}}} \mathcal{U}_j^E + \sum_{l:K_l \in \mathcal{L}_{\mathbb{K}}} \mathcal{U}_l^K \tag{16}$$

$$= \sum_{j:E_j \in \mathcal{L}_{\mathbb{E}}} M_j - \sum_{l:K_l \in \mathcal{L}_{\mathbb{K}}} \mathcal{C}_l \tag{17}$$

The fault-tolerant metric $\pounds_j$ is designed to capture the effect of demand uncertainty in MEC platform. It depends on the approximated demand and actual demand of IoT devices. Formally we have:

$$\pounds_j = \begin{cases} 1, & \mathcal{G}_j(t+1) \approx \nu_m, \\ \Re, & \text{otherwise}, \end{cases} \tag{18}$$

where $\Re$ is a small value designed to provide if approximated demand $\mathcal{G}_j(t+1)$ is not close to actual demand ν_m. The incentive mechanism for MEC is described in Definition 1. Here, we envision to model an optimal double auction mechanism, which maximizes the social-welfare and also provides satisfactory demands for all IoT devices.

Social-Welfare† Maximization. We define a winner selection scheme through a double-auction social-welfare maximization approach using integer linear programming (ILP). Formally:

$$\text{Maximize} \quad \pounds_j \mathcal{G}_j(t) \Big[\sum_{j:E_j \in \mathcal{L}_{\mathbb{E}}} y_j \Omega_j - \sum_{l:K_l \in \mathcal{L}_{\mathbb{K}}} z_l \Theta_l \Big], \tag{19}$$

$$\text{Subject to} \quad G_j \leq G_{th}, E_j \in \mathbb{E} \tag{20}$$

$$SCT_i > SCT_{th}, \forall i \tag{21}$$

$$\mathcal{C}_l \geq \mathcal{C}_{th}, K_l \in \mathcal{K} \tag{22}$$

$$\Theta_l, \Omega_j \in \{0, 1\}, \forall j \tag{23}$$

†The social welfare is defined as the summation of the designed platform's profit levels and all IoT devices' and servers' utility values.

Constants. The platform captures several parameters as inputs: set of services $\bar{S}$, set of edge servers $\mathcal{K}$, bidding profile of IoT devices' and servers' $\boldsymbol{y}$ and $\boldsymbol{z}$, concerned service profile set of edge servers' $\boldsymbol{S} = \{S_1, S_2, \cdots, S_N\}$, probability of successful execution λ_j^i.

Variables. Here, $\boldsymbol{\Omega} = (\Omega_1, \Omega_2, \cdots, \Omega_L)$ denotes the binary factors for incentive mechanism. Here $\Omega_l = 1$ denotes that service S_i will be executed, and thus, IoT device E_j is a winning IoT device (i.e., $E_j \in Y_E$), whereas $\Omega_i = 0$ denotes $E_j \notin Y_E$. Similarly, the mechanism considers another vector of K binary variables, $\boldsymbol{\Theta} = (\Theta_1, \Theta_2, \cdots, \Theta_K)$, where $\Theta_l = 1$ indicates that edge server K_l is a winning edge server (i.e., $K_l \in Y_K$), and $\Theta_l = 0$ means $K_l \notin Y_K$.

Objective. The designed objective function satisfies the following $\sum_{j:E_j \in \mathcal{L}_{\mathbb{E}}} y_j \Omega_j - \sum_{l:K_l \in \mathcal{L}_{\mathbb{K}}} z_l \Theta_l = \sum_{j:E_j \in Y_E} y_j \Omega_j - \sum_{l:K_l \in Y_K} z_l \Theta_l$ which is exactly the social welfare defined in Definition 5 based on the IoT devices' and servers' bids.

Constraints. Equation (19) describes the objective function and Equation (20) represents that the demand response of an IoT device G_j is to be greater than the threshold demand response G_{th}. The service completion time SCT_i should be higher than the threshold service completion time SCT_{th} as in Equation (21). The total execution cost $\mathcal{C}_l$ should be higher than the threshold execution cost $\mathcal{C}_{th}$ as represented in equations (20), (21), and (22). We get the solution of optimization problem using the Lagrangian method [23].

Algorithm 1 Demand Approximation: A Heuristic Approach

Inputs: IoT devices ($\mathbb{E}$), services $\mathbb{S}$ and time period $\bar{\tau}$.
Output: Optimal demand $\mathcal{G}_j(\bar{\tau}+1)$ and waiting time τ_{wait}.

Initialize $\tau_{wait} = 0$.
Initialize $\mathbb{E} = E$ and $\mathbb{S} = S$.
for each edge user associated with IoT device E_j **do**
  if $\bar{\tau} < \tau_{wait}$ **then**
    First, quantify the demand response profile $\mathcal{F}$.
    Estimate the rate of change in demand response ζ_j.
    Predict the optimal demand response, $\mathcal{G}_j(\tau+1)$.
    if $\mathcal{G}_j(\tau+1) \geq \mathcal{G}_{th}$ **then**
      Calculate resource utilization factor $\eta_j(\tau)$.
      Estimate the service flow $f_j^t(x')$.
      Get the actual resource requirement ν_j^t.
      Update waiting time $\tau_{wait} = \tau_{wait}$.
    else
      Updated set of IoT devices $\mathbb{E} = \mathbb{E}$.
      Non-optimal resource requirement $\widehat{\nu_j^\tau}$.
      Update waiting time $\tau_{wait} = \tau_{wait} + 1$.
    end if
  end if
end for
Return $\mathcal{G}_j(\bar{\tau})$, $\nu_j^{\bar{\tau}}$ and τ_{wait}.

A. Algorithm Design

We attribute our approach *DECADE*, as in: fault-tolerant mechanism **D**esign for **E**dge **C**omputing IoT Networks with **DE**mand uncertainty. It comprised of two algorithms: *Fault-Tolerant Mechanism Design* and *Heuristic Service Scheduling*.

Proposition 1. *The DECADE scheme, formalized with problem (19) is NP-hard.*

Proof. We show the NP-hardness of DECADE from the Uncapacitated Facility Location (UFL) problem [24], considered as a NP-hard problem using polynomial-time reduction. The recreation of the UFL problem should be designed while considering the unit price and demand of edge users. The demand approximation and fault-tolerant mechanism of DECADE should be mapped with the UFL problem and can be defined as a NP-hard problem. For this, we set the other parameters in DECADE to 0. □

Equation (8) represents the main optimization problem and hence we can get the solution by applying the interior point algorithm (IPA). The main motive is to optimize $\mathcal{G}_j(t)$ applying the Lagrangian multipliers (LM). The gradient decent method has been used for the solution of DECADE. Thereafter, we model a heuristic method to find the optimal solution globally observing local values in each step. Small and moderate-sized sample values of the ILP in (19) should be solved in polynomial time, applying existing solvers like Gurobi [25]. In this process, with higher values, the ILP can attain reasonably higher time to get into converge point, it is not suitable for fault-tolerant mechanism design under demand uncertainty. Therefore, we present a heuristic solution to design a fault-tolerant mechanism.

Algorithm 2 Fault-Tolerant Mechanism Design

Inputs: IoT devices ($\mathbb{E}$), services $\mathbb{S}$, $\mathcal{G}_j(\bar{t})$, $\nu_j^{\bar{t}}$ and τ_{wait}.
Output: Optimized fault-tolerant value $\bar{\mathcal{L}}_j$.

Define $\tau_{wa} = \tau_{wait}$.
Define $\mathcal{Z} = n$, $A = m$.
Consider the fault-tolerant metric.
for IoT devices E_j **do**
  if $\tau_{wait} < \tau_{wa}$ **then**
    Design utility function $\mathcal{U}_j^E$ for IoT device E_j.
    Design utility function $\mathcal{U}_i^K$ for an edge server K_l.
    Calculate profit level $\mathbb{P}_{le}$ for the designed platform.
    Estimate fault-tolerant metric $\mathcal{L}_j$.
    if $\mathcal{U}_j^E \geq \mathcal{U}_{th}$ and $\mathcal{U}_i^K \geq \mathcal{U}_{th}$ **then**
      Updated set of IoT devices $\bar{\mathbb{E}} = \mathbb{E} \cap E_j$.
      Optimized fault-tolerant value $\bar{\mathcal{L}}_j$.
      Revise $\tau_{wa} = \tau_{wa}$.
    else
      Updated set of IoT devices $\mathbb{E} = \mathbb{E}$.
      Compute SCT ($\widehat{\mathcal{L}_j}$) (non-optimal).
      Revise $\tau_{wa} = \tau_{wa} + 1$.
    end if
  end if
end for
Return $\bar{\mathcal{L}}_j$.

We discuss the heuristic algorithm for demand approximation of IoT devices. In Algorithm 1, we first quantify the demand response profile $\mathcal{F}$ and estimate the rate of change

in demand response ζ_j. Thereafter, we quantify the optimal demand, $\mathcal{G}_j(\tau+1)$. If the predicted value $\mathcal{G}_j(\tau+1) \geq \mathcal{G}_{th}$ is higher than the threshold, then we calculate the resource utilization factor $\eta_j(\tau)$ and estimate the service flow $f_j^{\tau}(x')$. Also, we get the actual resource requirement ν_j^{τ}. As depicted in Algorithm 2, initially, we should fed the required inputs, i.e., IoT devices ($\mathbb{E}$), services $\mathbb{S}$ and τ_{wait}. Since, the optimization problem is considered to be NP-hard, hence we provide a fault-tolerant heuristic method to minimize the SCT and price. First, we define $\tau_{wait} = 0$. Then, we operate the algorithm for each IoT device E_j. If the waiting time τ_{wait} is lower than the average time τ_{wa}, i.e., $\tau_{wait} < \tau_{wa}$, then the utility function $\mathcal{U}_j^E$ for IoT device E_j and utility function $\mathcal{U}_i^K$ for an edge server K_l should be computed. Afterward, we estimate profit level $\mathbb{P}_{le}$ for the designed platform and calculate the fault-tolerant metric $\pounds_j$. If the utility functions $\mathcal{U}_j^E$ and $\mathcal{U}_j^K$ are higher than the mean threshold utility $\mathcal{U}_{th}$, then the set of IoT devices $\bar{\mathbb{E}} = \mathbb{E} \cap E_j$ should be updated. The average time τ_{wa} should also be updated. Then, we get the optimal fault-tolerant value $\bar{\pounds}_j$ using ILP. The algorithm should be stop compiling if the waiting time surpasses the maximum time τ_{wa}^{max}.

Table I: Experimental Parameters

Parameter	Value
Bandwidth capacity	20 MHz
CPU cycles count for each task	1,000 Megacycles
Deadline for each service	[4000, 6000] ms
Resource demand of each service	[10, 20] MHz
Transmit power of IoT device	100 mWatts
Capability of IoT device	0.85 GHz
Capability of the MEC server	100 GHz
Arrival rate following Poisson process	[0, 10] unit/sec
IoT traffic flow size	100 Mbits
Service arrival distribution	[0, 10],

Proposition 2. *The time complexity of DECADE can be $O(\mathcal{K}n^2)$, n represents the count of IoT devices.*

Proof. In the beginning, IoT devices approximate their optimal demand response to minimize the overall SCT. The demand approximation method is computed for n rounds with complexity $O(\mathbb{B}n^2)$. Thereafter, we proceed for the fault-tolerant mechanism, in which, the IoT devices optimizes the fault-tolerant rate using a fault-tolerant auction mechanism. The proposed fault-tolerant mechanism is computed for n rounds in the presence of multiple IoT devices. The complexity of fault-tolerant algorithm is computed as $O(\mathbb{R}n)$. Both these methods provide us the total computational complexity, we get, $T(n) = \mathbb{L}_1\{\mathbb{B}T(n^2) + \mathbb{R}T(n)\} + \mathbb{L}_2 T(1)$. After joining both the methods, we can compute the total complexity as $O(\mathcal{K}n)$, where $\mathcal{K} = \mathbb{B} + \mathbb{J}$. Thus, the proposed DECADE algorithm has complexity of $O(\mathcal{K}n^2)$. □

V. DECADE Evaluation

In this section, we analyze the performance of DECADE, while varying different networking parameters. Our evaluation metrics are cost, latency, service execution, SCT, overhead, and fault-tolerance.

A. Simulation Settings

We enumerated all the experimental parameters in this section as depicted in Table I. With align to previous paper [26], we use an Arduino kit and an Intel i5 CPU to represent the IoT device to get the practical usefulness of computational capabilities. We consider a set up, where we have 100 edge users dispersed over a region of 5.5 km × 5.5 km. Each edge user has 10 tasks, and each task of an edge user can only select one edge server for computational execution. The deadline of a task is considered to be within 150ms to 10s. We consider that each edge user is trying to execute an application of machine learning inference, the size needed for each task to be varied in between 500 KB to 5 MB, and the number of CPU cycles to be computed is 1 GHz. Similarly, we consider that there are 20 edge servers, the system bandwidth of the edge server is 25 MHz. To configure an edge server, we consider the mean processing operation is 15 Million Instructions Per Second (MIPS) and the capacity of a general cloud provider is considered to be 1500 MIPS. The MEC servers are situated close to a base station (BS). The compute potential of MEC server is assigned to 130 GHz and the computational potential of IoT device is assigned to 0.85 GHz. The communication backhaul latency factor is set to 0.0005 sec/KB [27], [28]. The offloading duration of services to IoT devices are chosen within $7-14$ ms. The size of services are considered to be within the scale $500-950$ KB. The latency demand of IoT devices is set to be within the scale $0.8-1.2$ s. The IoT devices communicates to edge servers following wireless communication (i.e., 5G, WiFi and ZigBee). On the other hand, the edge servers communicates to cloud through broadband connection. We consider that the number of tasks that an edge server can compute is $5-7$, and that the transmit power of IoT device is 300 Mw. To consider the topological connection, we contemplated AttMpls topology borrowed from the Internet Topology Zoo [29]. Besides that, to generate the IoT traffic flow, we considered the D-ITG traffic generator [30] from the real-time traces as discussed in [31]. For our method, we generally allocate the resources to tasks in an edge server with priority based on their applications, and redirect the other tasks to the cloud if the edge platform failed to provide required resource to other tasks within its deadline.

Evaluation Metrics. The first metric that we consider is the average service delay, defined as the average time needed to process and execute the edge services generated by IoT devices. The second metric that we consider is the service utilization or service completion time (SCT), calculated using the count of successful execution of services and the count of available services.

$$\text{SCT} = \frac{\text{Count of successful execution of services}}{\text{Count of available services}}$$

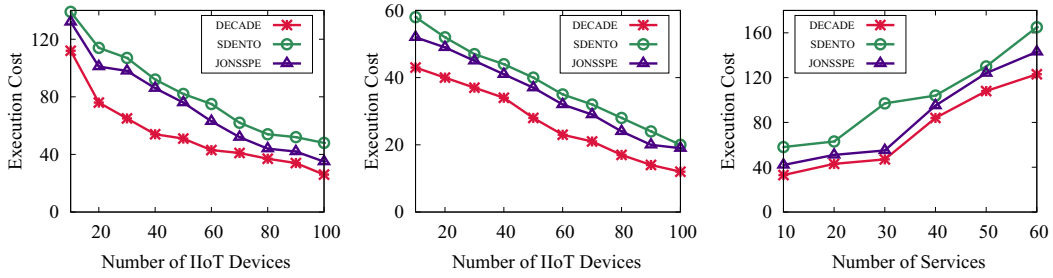
An higher value of SCT implies better performance. Other performance metrics that we consider are fault-tolerance and execution cost.

Benchmarks. To evaluate the performance of our approach, we compared it against two benchmark algorithms – *JONSSPE* [32] and *SDENTO* [33]. *JONSSPE* [32] considered a joint optimization of network and service placement in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3075681, IEEE Internet of Things Journal

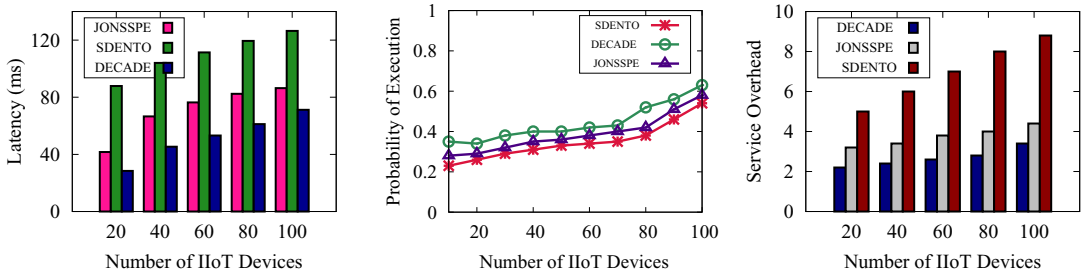
Table II: Experimental Settings

Settings	No. of Users	No. of Services	Mean Cost	Prob. of Exe. (PoE)
I	100	15	15	0.6
II	100	$[10-50]$	15	0.6



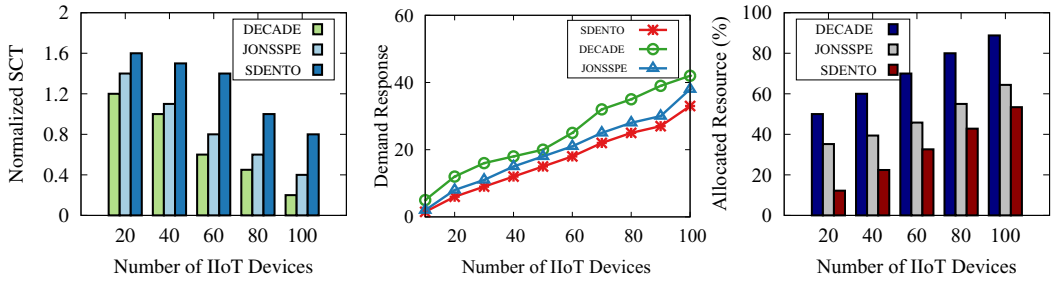
(a) Execution Cost (Single-Service) (b) Execution Cost (Multi Service) (c) Total Execution Cost

Figure 2: (a)-(c) Analysis of execution costs of single-service and multi-service mechanisms. The single-service means the IoT device has only one task to execute and multi-service means the IoT device has multiple tasks to execute.



(a) Service Latency (b) Prob. of Successful Execution (c) System Overhead

Figure 3: (a) Performance analysis of latency with the varying number of IoT devices. (b)-(c) Analysis of probability of successful execution and service overhead with varying numbers of IoT devices.



(a) Normalized SCT (b) Demand Response (c) Allocated Resource

Figure 4: (a) Analysis of SCT with the varying number of IoT devices. (b) Analysis of demand response with varying numbers of IoT devices. (c) Analysis of allocated resource with varying numbers of IoT devices.

MEC to minimize the switching latency and maximize the QoS. Initially, the authors design an online algorithm to solve the proposed NP-Hard optimization problem. *SDENTO* [33] instead investigates the task offloading algorithm for SDN-enabled ultra-dense network. The basic motive of this algorithm is to optimize the average latency and energy efficiency of user devices. To do so, the authors design an optimal task offloading algorithm using an NP-Hard mixed integer non-linear program (MINLP). They problem is split into a task assignment and a resource distribution sub-problem. By searching for an optimal values from these two parts, they get the increased performance. Similarly to our benchmark approaches, our fault-tolerant mechanism, detailed in Algorithms 1 and 2 considers demand uncertainty takes into account delay, offloading priority, completion time and mobility.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2021.3075681, IEEE Internet of Things Journal

B. Results and Discussion

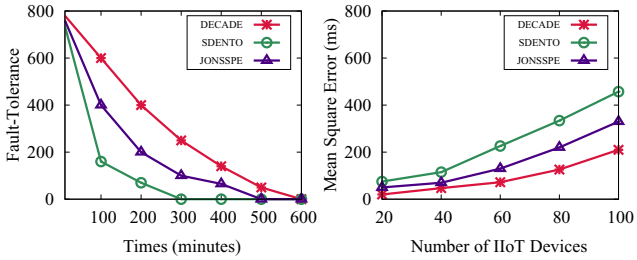
In this section we present our evaluation results. The parameters of our simulations are shown in Table II.

1) Cost Analysis: First, we study the execution cost derived from the execution of single and multiple services. From Figure 2(a), we observe that the execution cost degrades sharply, as the count of edge users increases and then changes steadily. As the costs follow the same kind of probabilistic distribution, hence newly joined edge users may not help in getting optimal average execution cost. We can observe that DECADE optimally approximates the demand response of edge users, thus our mechanism works better than both the SDENTO and JONSSPE scheme. Next, we evaluate the execution cost for multiple services defining different count of edge users and services using the experiment settings in Table II and depicts in Figure 2(b) and Figure 2(c). We notice that the execution cost degrades as the count of edge users increases in Figure 2(b). As the count of edge users can be large enough in a time period, hence the execution cost should converge and minimal overtime. If we consider an optimal economical market, the platform can lure more edge users providing optimal prices, and lower the execution cost to provide a stringent delay to users. In Figure 2(c), the execution cost increases as the count of executed services increases, as we need to accommodate a higher number of edge users. As the approximation ratio of DECADE is large, hence the execution costs are considered to be optimal. DECADE provides better resource distributions among available tasks, which help them to execute within their deadlines. Hence, the cost is optimal for DECADE.

2) Latency Analysis: Figure 3(a) represents the latency incurred by the available services at the edge. We notice that the delay of the network increases as the count of edge users increases. As the number of edge users increases, then the edge users contending for executing their services may face congestion, which automatically increases the latency. We compare DECADE with the existing solutions and DECADE outperforms the other schemes by 17% and 24%, respectively. The existing schemes SDENTO and JONSSPE failed to provide optimal resources to tasks based on their demand profile, which in turn increases the end-to-end latency and violates service-level agreements. Whereas DECADE optimally approximates the resource demand of tasks and execute them within their prescribed deadlines. Hence, DECADE incurs relatively lesser delay than SDENTO and JONSSPE.

3) Execution Analysis: Figure 3(b) represents the PoE associated to edge services. We notice that the PoE increases with the number of edge users. As we approximate the demand response of edge users, we can associate the execution values optimally to edge users, which inherently increases the execution probability. However, we also compare DECADE with the existing solution, and found that DECADE outperforms them, although only by 4% and 5%, respectively.

4) Overhead Analysis: In our work, the overhead is defined as an amalgamation of excess resources required to execute the tasks at the edge. Figure 3(c) depicts the system overhead while differing the number of edge users. Using the figure, we can notice that the system overhead of DECADE increases



(a) Fault Tolerance (b) MSE

Figure 5: (a) Fault-tolerance of DECADE with respect to time. (b) Analysis of mean square error (MSE).

while differing the number of edge users. However, we observe that DECADE is able to provide a better system overhead to edge users with respect to other existing methods. Therefore, the scalability of the platform increases. On the other hand, the existing solutions failed to come up with optimal overhead to edge users. However, DECADE outperforms the existing solutions JONSSPE and SDENTO by 14% and 21%.

5) SCT Analysis: Figure 4(a) shows that the service execution time for a varying number of edge users. Due to expansion in the count of edge users, the services at the edge also multiply, hence the service execution time also increases naturally. Even in this case, DECADE outperforms the benchmark, as it can able to execute a large number of tasks within their deadlines. Whereas the existing solutions failed to provide such guarantees, which in turn provides lesser SCT. Hence, the service execution time using DECADE is lesser than other approaches by 12-27%.

6) Demand Analysis: Figure 4(b) shows that the demand response of edge users. Due to an increase in users, the count of existing services that can be simultaneously supported also expands at the edge, it explicitly grows the demand response. DECADE outperforms the existing JONSSPE and SDENTO scheme, as our demand approximation scheme can accurately identify the required resources of IoT devices and thereafter we provide a fault-tolerant mechanism by which they can increase their bid. The demand response of IoT devices using DECADE is higher than other approaches by 15-23%.

7) Allocated Resource Analysis: Figure 4(c) shows that the allocated resources for a varying number of IoT devices. Due to the increase in IoT devices, the count of generated and executed services also increases, therefore resource exigency of services also increases. Even in this case, DECADE outperforms the benchmark. Hence, the service execution time using DECADE is lesser than other approaches by 32-47%.

8) Fault-Tolerance Analysis: Figure 5(a) presents the fault-tolerance ability of DECADE. We observe from the figure that DECADE provides better services and adaptive to network failures, demand uncertainty, link cuts, and device heterogeneity than the other approaches JONSSPE and SDENTO. Due to the auction mechanism used by DECADE, the fault-tolerant capacity of MEC expands with demand uncertainty. To evaluate this property, we compared our schema under a fault-tolerance metric. In particular, we observe the fault-tolerant calculated as the MSE. Figure 5(b) shows such MSE

of the DECADE, and illustrates that the MSE of the DECADE approach is lower compared to JONSSPE and SDENTO by 12% and 15%, respectively.

VI. Conclusion

In this paper, we presented DECADE, a novel incentive mechanism design for edge-based IoT networks to maximum social-welfare to edge users. Our approach uses a demand-approximation scheme, efficiently estimates the demand requirements of edge users, and minimizes the service execution cost. Furthermore, our scheme minimizes the execution cost of users. Extensive simulation results show that DECADE achieves better PoE and service delay than the existing approaches, such as JONSSPE and SDENTO. The service execution time using DECADE is lower than JONSSPE and SDENTO by 12-27%. Also, DECADE outperforms JONSSPE and SDENTO in terms of overhead by 14% and 21%. We believe that our solution opens other research in optimal and dynamic energy-efficient mechanisms for fault-tolerance in MEC-based IoT networks.

Acknowledgment

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.01-2019.322.

References

[1] L. Gu, D. Zeng, S. Guo, A. Barnawi, and Y. Xiang, "Cost-Efficient Resource Management in Fog Computing Supported Medical CPS," *IEEE Transactions on Emerging Topics in Computing*, vol. PP, no. 99, pp. 1–1, 2016.

[2] A. Samanta and J. Tang, "Dyme: Dynamic microservice scheduling in edge computing enabled iot," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6164–6174, 2020.

[3] 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.

[4] P. Zhang, Y. Zhang, H. Dong, and H. Jin, "Mobility and dependence-aware qos monitoring in mobile edge computing," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2021.

[5] T. G. Nguyen, T. V. Phan, B. T. Nguyen, C. So-In, Z. A. Baig, and S. Sanguanpong, "Search: A collaborative and intelligent nids architecture for sdn-based cloud iot networks," *IEEE access*, vol. 7, pp. 107 678–107 694, 2019.

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

[7] A. Samanta and Z. Chang, "Adaptive service offloading for revenue maximization in mobile edge computing with delay-constraint," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3864–3872, 2019.

[8] S. Chu, Z. Fang, S. Song, Z. Zhang, C. Gao, and C. Xu, "Efficient multi-channel computation offloading for mobile edge computing: A game-theoretic approach," *IEEE Transactions on Cloud Computing*, pp. 1–1, 2020.

[9] N. Wang, B. Varghese, M. Matthaiou, and D. S. Nikolopoulos, "Enorm: A framework for edge node resource management," *IEEE Transactions on Services Computing*, pp. 1–1, 2017.

[10] M. Hu, Z. Xie, D. Wu, Y. Zhou, X. Chen, and L. Xiao, "Heterogeneous edge offloading with incomplete information: A minority game approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 9, pp. 2139–2154, 2020.

[11] J. Xia, G. Cheng, D. Guo, and X. Zhou, "A qoe-aware service enhancement strategy for edge artificial intelligence applications," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[12] H. Xiong, Y. Wu, C. Jin, and S. Kumari, "Efficient and privacy-preserving authentication protocol for heterogeneous systems in iiot," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[13] Y. Liao, L. Shou, Q. Yu, Q. Ai, and Q. Liu, "An intelligent computation demand response framework for iiot-mec interactive networks," *IEEE Networking Letters*, pp. 1–1, 2020.

[14] R. Fantacci and B. Picano, "A matching game with discard policy for virtual machines placement in hybrid cloud-edge architecture for industrial iot systems," *IEEE Transactions on Industrial Informatics*, pp. 1–1, 2020.

[15] V. Casola, A. De Benedictis, S. Di Martino, N. Mazzocca, and L. L. L. Starace, "Security-aware deployment optimization of cloud-edge systems in industrial iot," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[16] Z. Zhao, Y. Shi, B. Diao, and B. Wu, "Optimal data caching and forwarding in industrial iot with diverse connectivity," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2288–2296, 2019.

[17] X. Lai, Q. Hu, W. Wang, L. Fei, and Y. Huang, "Adaptive resource allocation method based on deep q network for industrial internet of things," *IEEE Access*, vol. 8, pp. 27 426–27 434, 2020.

[18] Z. Zhou, S. Yu, W. Chen, and X. Chen, "Ce-iot: Cost-effective cloud-edge resource provisioning for heterogeneous iot applications," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8600–8614, 2020.

[19] Q. Yuan, J. Li, H. Zhou, T. Lin, G. Luo, and X. Shen, "A joint service migration and mobility optimization approach for vehicular edge computing," *IEEE Transactions on Vehicular Technology*, pp. 1–1, 2020.

[20] A. A. Al-Habob, O. A. Dobre, A. G. Armada, and S. Muhaidat, "Task scheduling for mobile edge computing using genetic algorithm and conflict graphs," *IEEE Transactions on Vehicular Technology*, pp. 1–1, 2020.

[21] 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*, 2019.

[22] A. Samanta, Y. Li, and F. Esposito, "Battle of microservices: Towards Latency-Optimal heuristic scheduling for edge computing," in *IEEE NetSoft*, 2019.

[23] O. Du Merle, J.-L. Goffin, C. Trouiller, and J.-P. Vial, "A Lagrangian Relaxation of the Capacitated Multi-item Lot Sizing Problem Solved with an Interior Point Cutting Plane Algorithm," in *Approximation and Complexity in Numerical Optimization*. Springer, 2000, pp. 380–405.

[24] J. Krarup and P. M. Pruzan, "The Simple Plant Location Problem: Survey and Synthesis," *European journal of operational research*, vol. 12, no. 1, pp. 36–81, 1983.

[25] I. G. Optimization, "Gurobi optimizer reference manual," Available: http://www.gurobi.com, 2016.

[26] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, "On reducing iot service delay via fog offloading," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 998–1010, 2018.

[27] 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*, 2016.

[28] A. Samanta, Z. Chang, and Z. Han, "Latency-Oblivious Distributed Task Scheduling for Mobile Edge Computing," in *IEEE GLOBECOM*, 2018, pp. 1–7.

[29] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.

[30] A. Botta, A. Dainotti, and A. Pescapé, "A tool for the generation of realistic network workload for emerging networking scenarios," *Computer Networks*, vol. 56, no. 15, pp. 3531–3547, 2012.

[31] A. Sivanathan, D. Sherratt, H. H. Gharakheili, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Characterizing and classifying iot traffic in smart cities and campuses," in *IEEE INFOCOM Workshops*, 2017, pp. 559–564.

[32] B. Gao, Z. Zhou, F. Liu, and F. Xu, "Winning at the starting line: Joint network selection and service placement for mobile edge computing," in *IEEE INFOCOM*, 2019, pp. 1459–1467.

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