A DRL-based service offloading approach using DAG for edge computational orchestration.

MEKALA, M.S., DHIMAN, G., SRIVASTAV, G., NAIN, Z., ZHANG, H., VIRIYASITAVAT, W. and VARMA, G.P.S.

2022

A DRL-Based Service Offloading Approach Using DAG for Edge Computational Orchestration

M. S. Mekala, Gaurav Dhima, Gautam Srivastav, *Senior Member, IEEE*, Zulqar Nain, Haolin Zhang, Wattana Viriyasitavat, *Senior Member, IEEE*, and G. P. S. Varma

***Abstract*— Edge infrastructure and Industry 4.0 required services are offered by edge-servers (ESs) with different computation capabilities to run social application's workload based on a leased-price method. The usage of Social Internet of Things (SIoT) applications increases day-to-day, which makes social platforms very popular and simultaneously requires an effective computation system to achieve high service reliability. In this regard, offloading high required computational social service requests (SRs) in a time slot based on directed acyclic graph (DAG) is an NP-complete problem. Most state-of-art methods concentrate on the energy preservation of networks but neglect the resource sharing cost and dynamic subservice execution time (SET) during the computation and resource sharing. This article proposes a two-step deep reinforcement learning (DRL)-based service offloading (DSO) approach to diminish edge server costs through a DRL influenced resource and SET analysis (RSA) model. In the first level, the service and edge server cost is considered during service offloading. In the second level, the R-retaliation method evaluates resource factors to optimize resource sharing and SET fluctuations. The simulation results show that the proposed DSO approach achieves low execution costs by streamlining dynamic service completion and transmission time, server cost, and deadline violation rate attributes. Compared to the state-of-art approaches, our proposed method has achieved high resource usage with low energy consumption.**

***Index Terms*— Adaptive quality-of-service (QoS), deep reinforcement learning (DRL) method, edge computing, optimal measurement analysis, service offloading (SO) and scheduling.**

Manuscript received November 15, 2021; revised January 26, 2022; accepted March 18, 2022. This work was supported in part by the Basic Science Research Programs of the Ministry of Education under Grant NRF-2018R1A2B6005105 and in part by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MSIT) under Grant 2019R1A5A8080290. *(Corresponding author: M. S. Mekala.)*

M. S. Mekala is with the Department of Information and Communication Engineering, Yeungnam University, Gyeongsan 712-749, South Korea, and also with RLRC for Autonomous Vehicle Parts and Materials Innovation, Yeungnam University, Gyeongsan 712-749, South Korea (e-mail: msmekala@yu.ac.kr).

Gaurav Dhiman is with the Department of Computer Science, Government Bikram College of Commerce, Patiala 147001, India, also with the University Centre for Research and Development, Department of Computer Science and Engineering, Chandigarh University Gharuan, Mohali 140413, India, and also with the Department of Computer Science and Engineering, Graphic Era Deemed to be University, Dehradun 248002, India (e-mail: gdhiman0001@gmail.com).

Gautam Srivastava is with the Department of Math and Computer Science, Brandon University, Brandon, MB R7A 6A9, Canada, and also with the Research Center for Interneural Computing, China Medical University, Taichung 40402, Taiwan (e-mail: srivastavag@brandonu.ca).

Zulqar Nain is with the Department of Information and Communication Engineering, Yeungnam University, Gyeongsan 712-749, South Korea (e-mail: zulqarnain@ynu.ac.kr).

Haolin Zhang is with the Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH 43210 USA (e-mail:zhang.10749@osu.edu).

Wattana Viriyasitavat is with the Department of Statistics, Faculty of Commerce and Accountancy, Chulalongkorn University, Bangkok 10330, Thailand (e-mail: hardgolf@gmail.com).

G. P. S. Varma is with the Department of Computer Science, and Engineering, Koneru Lakshmaiah Education Foundation, Vijayawada, Andhra Pradesh 522502, India (e-mail: gpsvarma@gmail.com).

I. Introduction

SOCIAL edge service (SES) is an emerging service mechanism in the Social Internet of Things (SIoT) orchestration for user-centric reliable communication and computation. Natural Resources Defense Council (NRDC) report confess that an increased quantity of data centers release 100 million metric tons of carbon dioxide (CO_2) by 2022 [1], [2] and in future, it increases due to 5G network implementation with futuristic visions of 6G. Usually, data centers share Petabytes (PB) of data per day, impacting the bandwidth cost of internet service providers (ISPs). However, state-of-the-art approaches neglect heterogeneity of service request while measuring bandwidth cost of service provider and server energy usage cost also differ because of geometric-interconnection among data centers [3]. If the service execution rate (SER) is increased, then the resource usage rate (RUR) increases certainly (SER $\propto$ RUR). Service queue length (SQL) and current resource capacity of edge-servers (ESs) have not been examined during service offloading (SO) [4] since the service request device may move out of range of the responding server. Therefore, to avoid service delay the design of SO strategy is certainly essential. In our work, the **R-retaliation** method is designed to optimize SO by reducing offloading time, execution cost, adaptive resource utilization, and the waiting queue length. The deep reinforcement learning (DRL)-based SO (DSO) approach measures the resource provision (RP) rate of the arrived services through the ES current status based on prognosticate service execution time (PSET) method. The article contributions in this article are as follows.

1) Develop a DSO approach to reduce subservice execution time (SET) cost, transmission time and optimize energy usage rate.
2) Develop an R-retaliation analysis model to optimize the RP rate, service deadline violation rate, and SET fluctuations based on prognostic big-data evaluation factors.
3) Develop prognosticate execution cost method to regulate the service execution time fluctuations.
4) Develop adaptive methods to evaluate.
    a) Service request transmission time.
    b) SER.
    c) Energy preservation method.

The rest of the article is organized as follows. Section II briefly explains research gaps and problems of extant approaches. Sections III and IV describes the proposed system and its mathematical models. Section V describes our proposed algorithm in detail.

II. Related Work

The service allocation issue is treated as a NP-complete problem because the tasks are recursively assigned more than one time to machines. In Wang [5] and Mekala *et al.* [6], the authors have concentrated on the same issue with proof of comprehensive research method, and it is a delicate mechanism because of the substantial number of services with various resources. In [7], intelligent offloading method (IOM) has been designed for effective service scheduling. The authors initially evaluated the SET and RUR values which are not optimal. In [8], the quality-of-service (QoS)-aware cloud framework enables a task scheduling (TS) method called the QoS-Min-Min approach, and it estimates the resources for SO based on requests, but it has an inadequate execution time than traditional approaches. In [9], multiservice task computing offload algorithm (MTCOA) has been designed based on an evaluation of system cost to minimize offloading decisions of mobile edge computing (MEC). Still, the lengthy services have miss-mapped to low resource capacity ESs that influence the system's execution makespan (MS) and cost. In [10], Randomized Online Stack-centric Scheduling algorithm (ROSA) designed for effective service allocation based on cost-effective resource usage analysis. In [11], `task-graph template` has been designed for TS called Nondominated Sorting Genetic Algorithm (NSGA). However, the static system influence the system performance, and It would consider as further research—this drawback is streamlined by considering network device computation stability to meet QoS. In [12] and [13], the service deadline and cost attributes are considered to select the ES for effective service scheduling. In [14]–[16], a new scheduling approach called Preference-Inspired Coevolutionary Algorithms (PICEA-g) is designed based on graph theory, where cost and length of the service queue are considered. However, this approach did not allow the resubmission of failed tasks. In [17] and [18], the workload scheduling streamlines the issues caused by uncertain machine availability.

In [19], the service allocation method subtracts the duplicate services that influence the SO ratio and system performance. Our articles main objective is to develop an adaptive SO policy by considering SET, energy utilization rate (EUR), and RUR with a balanced workload.

III. System Model and Problem Formulation

Fig. 1 exhibits a task mapping system with multidimensional coordinates. Offloading the service to the suitable server plays an important role to meet the QoS, which can observe in Fig. 1(a). The service arrival rate from various platforms may require several computing resources for its execution to meet the deadline as mentioned in Fig. 1(b). For instance, $t_i^{j,L}$ refers the service from workLoad (L), has to assign jth ES for its effective execution.

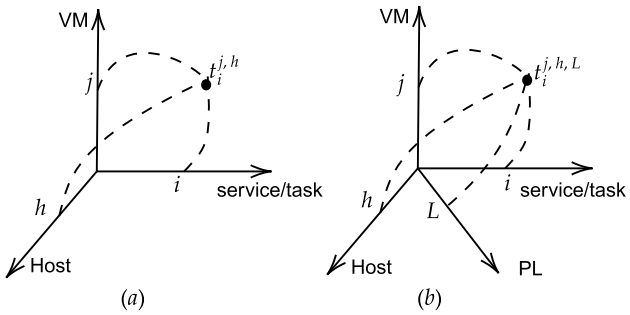


Fig. 1. 3-D underlying objective analytic mechanism. (a) Multidimensional objective analysis mechanism. (b) Core objective mechanism.

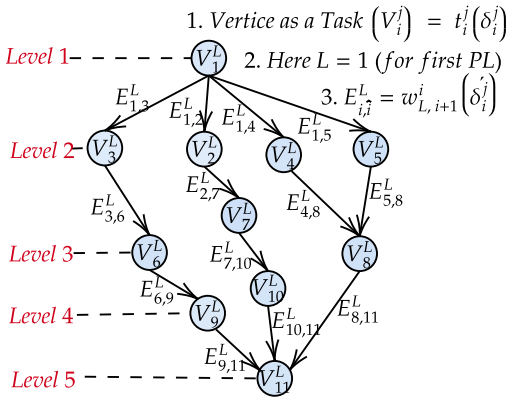


Fig. 2. DAG graph model.

A. DAG Model

The leased ES represented as $\mathbb{Q} = V_j | j \in [1, 2, \ldots]$. Here, V_j refers to the jth ES. The server cost and its resource capacity rate $C(\mathrm{ES}_j) \propto \Omega_j$ are proportionally dependent on each other. During the evaluation of cost, the service provider applies the charges for storage service, but the storage instance has not been considered since our resource consolidation scheme concentrates on SO. Therefore, the storage space cost has been neglected in our mechanism. The cost of communication to transfer the data is considered during ES resource consolidation. Scheduling high-rank ES might reduce the response time of the system, but it causes high tenant costs. To formulate this issue, the below computation model considers ES rank and its overall weight rate. The weight factor diminishes the execution period of ES as given in the following equation:

$$\mu\left(t_{\mathrm{ic}}^j\right) = t_{\mathrm{ic}}^j \Big/ \mathrm{Min}\left(t_{\mathrm{ic}}^j\right). \tag{1}$$

Application workload or Program workloads (PLs) requires different computation resource sizes and different ranking ES to execute the service to meet QoS and service level agreement (SLA) bounded range. We epitomize the PL as $\aleph = (F_1, F_2, \ldots, F_L)$. Each PL enabled with three factors, which are $F_L = \{\mathrm{AT}_L, D_L, G_L\}$. Where AT_L, D_L refers arrival period, deadline period of individual PL. Now, G_L refers directed acyclic graph (DAG) graph, it enables four constructive parameters $G_L = \{T_L, E_L, \Gamma_L, W_L\}$, where $T_L = \{t_{1,L}, t_{2,L}, t_{3,L}, \ldots, t_{n,L}\}$ number of services, E_L denote edges between two services, Γ_L refers constructive time for service execution with normalized *packing policy* such as $\delta_{i,c}^j \in \Gamma_L$ and W_L refers adaptive weight time to transfer the data in

between two integrated services $(w^j_{i,\widehat{i}})$ such as $i, \widehat{i}$ and $w^j_{i,\widehat{i}} \in W_L$. In case, if two services are assigned to single ES, then the adaptive data transfer time weight factor is zero, i.e., $w^j_{i,\widehat{i}} = 0$. Fig. 2 shows 11−service DAG graph, here V^L_1 refers entry service request and V^L_{11} is exist service request. $E^L_{1,3}$ refers weight of data transfer time. $\Pr(t^j_{L,i}) = V^L_1$ and $\mathrm{Su}(t^j_{L,i}) = V^L_7$ are the service request successors and predecessors or vertex V^L_2.

B. Problem Formulation

The proposed system regulates the PL cost while accomplishing the targeted deadlines of SLA. Initially, the PL MS estimates with (2) by subtracting the PL arrival time from the maximum SET of among all service requests

$$\mathrm{MS}_L = \max_{t^j_{i,L} \in T_L} \left\{ \alpha\left(t^j_{\mathrm{ic}}\right) \right\} - \mathrm{AT}_L \tag{2}$$

$$\mathrm{Min} \sum_{L=1}^{F_L} \sum_{j=1}^{\mathrm{ES}_j} \sum_{i=1}^{T_i} \mathbb{Q} \times P_C \times \varpi^j_{i,L} \tag{3}$$

where we define arrival time AT_L of services, $\varpi^j_{i,L}$ refers weight value, ample of service completion time $\alpha(t^j_{\mathrm{ic},L})$, respectively. During the consolidation process, the below-mentioned conditions are essential considerations to accomplish the target. Note that, in this example, each ES assign receives on service request for effective execution, which evaluates with (4) and summation of all edge device (ED) capacity should not violate the capacity of its server; it estimates with the following equation:

$$\left.\begin{array}{l} \cup_{\mathrm{ES}_j \in h_m} \mathrm{ES}^r_j(h_m) = \mathrm{ES}^r_{j,m} \\ \mathrm{ES}^r_{i=1,m=1} \cap \mathrm{ES}^r_{j=1,m=2} = \emptyset \end{array}\right\} \quad \forall 1 \leq i, \quad j \,\&i \neq j \tag{4}$$

$$\mathrm{FVT}^X_{j,m} \;\leq\; \sum_{j=1}^{\mathrm{ES}_j} \mathrm{ES}^r_{j,m}(r_c) \;\leq\; h_m(r_c) \quad \forall\, 1 \leq j \leq m. \tag{5}$$

The summation of MS MS_L and arrival time AT_L of PL should be less than the deadline of the respective PL and is calculated as $\mathrm{MS}_L + \mathrm{AT}_L \leqslant D_L$. $\varpi^j_{i,L}$ refers weight value, which ensures the service request status. It should be less than or equal to 1 as $\sum_{j=1}^{\mathrm{ES}_j} \varpi^j_{i,L} \leqslant 1$. The service request executed by ES with ample of completion time $\alpha(t^j_{\mathrm{ic},L})$ and starting time $\beta(t^j_{\mathrm{is},L})$. Here, the summation of SET and its data transfer time $w^{j,L}_{i,\widehat{i}}$ are must be less than or equal to the starting time. It estimates with the following equation:

$$\alpha\left(t^j_{\mathrm{ic},L}\right) + w^{j,L}_{i,\widehat{i}} \leqslant \beta\left(t^j_{\mathrm{is},L}\right) \tag{6}$$

$$\varpi^j_{i,L} = \begin{cases} 1, & \text{if the Errand remain assign to } j\text{th ES.} \\ 0, & \text{Otherwise.} \end{cases} \tag{7}$$

IV. DSO Approach

A. PL Allocation Framework

The DSO approach concentrates on provisioning the entail resources to ES for executing the PL which can be observed in Fig. 3. A user allows submitting the service requests with various deadlines, while the other existing PL is under the execution process. The proposed framework enables *workload, resource, service manager, and PL pool sections*. The workload analyzer concentrates on dependency relations among

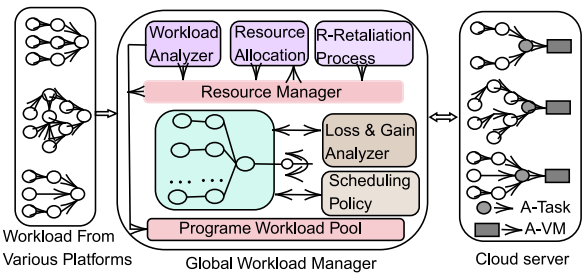


Fig. 3. PL allocation framework.

service request entities. The program workflow pool section accommodates unassigned service requests. The resource manager (RM) concentrates on regulating the computing resources dynamically toward the RP of liberating machines over the workload duty cycle. The service request manager is responsible for making offloading policy between service requests and ES with the coordination of the RM through the cost analysis model.

The SO process has been classified into three steps, namely *RP, Workload analysis, R-retaliation method*. The PL were initiated with two signals (service-request/PL arrival, service request completion signal). If the arrival signal has triggered, then the preestimation step starts to estimate priority through arrival time. The subworkload deadlines (s-deadlines) are evaluated based on previous outcomes, that should not violate the PL deadline. The $m/m/1$ model assigns the service requests to the suitable Es based on the resource usage analyzed by the RM. The rest of the unallocated service request (SR) are stored in the program workflow pool. The RP analysis step finalizes the suitable ES with the RM recommendation. This process impacts on reducing SET fluctuations of ES, except request waiting time on the same ES; because a single service request has been allocated at each time because the ES waiting time cost is less than the SET fluctuations cost of ES. The R-retaliation method estimates the priority of service requests as per the s-deadlines based on resource entail rate, RUR. This step plays an essential role in mitigating the SET.

B. Service Request Resource Measurement Method

The SR are listed in ascending order in a queue through the $m/m/1$ method. The below definitions streamline the SO process by assigning to a suitable ES.

Inference 1: If the service request is ready for execution, then initial service requests are executed or maybe it is the initial service request. Two essential steps have to satisfy before service request allocation.

1) Assessing the tentative SET (at what time the service request starts, its execution on the assigned ES).
2) Estimating whether the suitable ES is ready to accommodate the service requests at the specific time.

Thus, prognosticated starting time $\varphi(t^j_{i,L})$ calculates with (8), as a maximum value among current system time, prognosticated completion time, and maximum completion time of the service request

$$\varphi\left(t^j_{i,L}\right) = \mathrm{Max}\left\{\tilde{\lambda}, \hat{\alpha}\left(t^j_{\mathrm{ic},L}\right), \max_{t^j_{i,L} \in T_L} \left\{\alpha\left(t^j_{\mathrm{ic},L}\right), w^{j,L}_{i,\widehat{i}}\right\}\right\} \tag{8}$$

where $\hat{\lambda}$ and $\hat{\alpha}(t^j_{\text{ic},L})$ refers current system time, prognosticated completion time, respectively. $\alpha(t^j_{\text{ic},L})$ refers finish time of the service request on jth ES.

Theorem 1: The prognosticated execution time $(\hat{\varphi}(t^j_{i,L}))$ of service request on ES is equal to the product of two factors (weight of ES and weight of execution). Such that, $\hat{\varphi}(t^j_{i,L}) = \mu(t^j_{\text{ic}}) \times \phi(\delta^j_{i,c})$.

Proof: If sophisticated resources are available, then the service request execution time is $\delta^j_{i,c} \cong \Delta(\chi^j_{\text{ic}}, (\partial^j_{\text{ic}})^2)$. Where, $\phi(\delta^j_{i,c})$ is an approximate weight estimated with the following equation:

$$\phi(\delta^j_{i,c}) = \begin{cases} \chi^j_{\text{ic}} + \sqrt{\partial^j_{\text{ic}}}, & \partial^j_{\text{ic}} \Big/ \chi^j_{\text{ic}} \leqslant 1 \\ \chi^j_{\text{ic}} \times \left(1 + 1 \Big/ \sqrt{\partial^j_{\text{ic}}}\right), & \text{Otherwise} \end{cases} \tag{9}$$

where χ^j_{ic} expected value and ∂^j_{ic} is its variance with normal offloading policy $\Delta(\chi, \partial^2)$. This normalized equation is multiplied with constant variable.

Hence, $\Theta \times \delta^j_{i,c} \cong \Delta(\Theta \times \chi^j_{\text{ic}}, \|\Theta\| \times (\partial^j_{\text{ic}})^2)$. The resultant value is $\mu(t^j_{\text{ic}}) \times \delta^j_{i,c} \cong \Delta(\mu(t^j_{\text{ic}}) \times \chi^j_{\text{ic}}, \mu(t^j_{\text{ic}}) \times (\partial^j_{\text{ic}})^2)$. Thus, $\mu(t^j_{\text{ic}}) \times \delta^j_{i,c} + \mu(t^j_{\text{ic}}) \times \partial^j_{\text{ic}} = \mu(t^j_{\text{ic}})(\delta^j_{i,c} + \partial^j_{\text{ic}})$.

Since, $\therefore \phi(\delta^j_{i,c}) = \delta^j_{i,c} + \partial^j_{\text{ic}}$. Therefore, $= \mu(t^j_{\text{ic}}) \times \phi(\delta^j_{i,c})$. Hence, $L.H.S = R.H.S$, $(\hat{\varphi}(t^j_{i,L}) = \mu(t^j_{\text{ic}}) \times \phi(\delta^j_{i,c}))$.

Subsequently, the prognosticated execution time $\hat{\alpha}(t^j_{\text{ic},L})$ is equal to summation of prognosticated start time $\varphi(t^j_{i,L})$ and prognosticated completion time $\hat{\varphi}(t^j_{i,L})$.

Such that, $\hat{\alpha}(t^j_{\text{ic},L}) = \varphi(t^j_{i,L}) + \hat{\varphi}(t^j_{i,L})$. ■

Inference 2: Eligible ES should satisfies the below-listed conditions, i.e., which ES can successfully execute the assign service request to meet its deadline.

1) For service request execution, the augmented ES cost must be less, to meet SLA constraints. It estimates with $\hat{\varphi}(t^j_{i,L}) = \mu(t^j_{\text{ic}}) \times \phi(\delta^j_{i,c})$ along with satisfying (4).
2) In case, more than one ES satisfies (9), then less idle time of ES remain selects to balance the system RUR.

The feasible ES selection accomplishes through the heuristic technique by limiting the ES execution cost with enhanced RUR. As an indication of the service request deadline, the RM starts assigning the resources to all active ESs, which satisfies (9). In case, more than one ES satisfies (9), then the RM initiates to select another ES. It does not mean a violation of the individual service request deadline and it is equal to the violation of the whole PL deadline.

C. Preestimation and R-Retaliation Mechanism

This section is firmly the heart of this framework because primary attributes assessment is an essential step while making a decision. This section concentrates on estimating individual service request deadline called *s-deadline*, where *starting time, current completion time* of each arrival PL has become phenomenal parameters for the PL allocation process. The listed parameters have been designed toward making a PL and SO decision.

Let S-deadline of service request $d_{i,L}$, $\therefore d_{i,L} \in D_L$ can be estimated by using the following equation:

$$d_{i,L} = \varepsilon(t^j_{\text{ic}})_{\text{ost}} + \frac{1}{2}\left(\frac{\varepsilon\left(t^j_{\text{ic}}\right)_{\text{oft}} - \varepsilon\left(t^j_{1c}\right)_{\text{ost}}}{\varepsilon\left(t^j_{Tc}\right)_{\text{oft}} - \varepsilon\left(t^j_{1c}\right)_{\text{ost}}}\right) \times \left(\varepsilon\left(t^j_{Tc}\right)_{\text{lct}} - \varepsilon\left(t^j_{1c}\right)_{\text{ost}}\right) \quad \therefore d_{i,L} \in D_L \tag{10}$$

where $\varepsilon(t^j_{1c})_{\text{ost}}$, $\varepsilon(t^j_{Tc})_{\text{oft}}$ refers initial and final service completion time, respectively. Now, original completion time is $\varepsilon(t^j_{\text{ic}})_{\text{oft}} = \varepsilon(t^j_{\text{ic}})_{\text{ost}} + \phi(\delta^j_{i,c})$.

The original start time $\varepsilon(t^j_{\text{ic}})_{\text{ost}}$ of the service is estimated by using the following equation:

$$\varepsilon(t^j_{\text{ic}})_{\text{ost}} = \begin{cases} \text{at}_L, & \text{Where} \quad \text{at}_L \in \text{AT}_L, \quad \therefore \ i = 1 \\ \text{Max}\left\{\varepsilon\left(t^j_{\text{ic}}\right)_{\text{ost}} + \phi\left(\delta^j_{i,c}\right) + w^{j,L}_{i,\widehat{i}}\right\}, & \text{Otherwise.} \end{cases} \tag{11}$$

The current completion time $\varepsilon(t^j_{\text{ic}})_{\text{ost}}$ of the service is estimated by using the following equation:

$$\varepsilon\left(t^j_{\text{ic}}\right)_{\text{lct}} = \begin{cases} d_{i,L}, & d_{i,L} \in D_L, \quad \therefore \ i = \text{sn}. \\ \text{Min}\left\{\varepsilon\left(t^j_{\text{ic}}\right)_{\text{lct}} + w^{j,L}_{i,\widehat{i}} - \phi\left(\delta^j_{i,c}\right)\right\} & \text{Otherwise.} \end{cases} \tag{12}$$

Confence 1: Paying attention to identify the individual service request deadline and which should be less than PL deadline; such that, $d_{i,L} \leqslant D_L$. Subsequently, the variance time $(\hat{\partial}^j_{\text{ic}})$ between deadline time $(d_{i,L})$ of task and original start time $\varepsilon(t^j_{\text{ic}})_{\text{ost}}$. It remains an estimate as

$$\hat{\partial}^j_{\text{ic}} = d_{i,L} - \varepsilon\left(t^j_{\text{ic}}\right)_{\text{ost}}. \tag{13}$$

The variance of each service request S-deadline $\hat{d}_{i,L}$ is calculate with the following equation:

$$\hat{d}_{i,L} = \begin{cases} \hat{\varepsilon}\left(t^j_{\text{ic}}\right)_{\text{ost}} + \hat{\partial}^j_{\text{ic}}, & \text{If } \hat{\varepsilon}\left(t^j_{\text{ic}}\right)_{\text{ost}} + \hat{\partial}^j_{\text{ic}} < \varepsilon\left(t^j_{\text{ic}}\right)_{\text{lct}} \\ \varepsilon\left(t^j_{\text{ic}}\right)_{\text{lct}}, & \text{Otherwise.} \end{cases} \tag{14}$$

Simultaneously, the variance in original start time $\hat{\varepsilon}(t^j_{\text{ic}})_{\text{ost}}$ and original finish time $\hat{\varepsilon}(t^j_{\text{ic}})_{\text{oft}}$ is estimated using the following equations:

$$\hat{\varepsilon}\left(t^j_{\text{ic}}\right)_{\text{ost}} = \hat{\varepsilon}\left(t^j_{\text{ic}}\right)_{\text{oft}} - \phi\left(\delta^j_{i,c}\right) - w^{j,L}_{i,\widehat{i}} \tag{15}$$

$$\hat{\varepsilon}(t_{\text{ic}})^j_{\text{oft}} = \text{Max}\left\{\alpha\left(t^j_{ic,L}\right) + \phi\left(\delta^j_{i,c}\right) + w^{j,L}_{i,\widehat{i}}\right\}. \tag{16}$$

D. Service Capacity Computational Methods

1) Service-Request Data Acquisition Time: While service requests are executed, assessing both expected resource rate, and SET are essential to reduce the cost. Asset assigning factor requires each ES bandwidth and RAM capacity. The data transfer time evaluates with the following equation:

$$t_{\text{tt}} = \sum_{j=1}^{V}\sum_{i=1}^{T} \varpi^j_i \times t_d \Big/ 0.125 \times \left(\upsilon_b \Big/ \upsilon_r\right) \tag{17}$$

where ϖ^j_i alludes asset assigned weight, υ_b alludes ES bandwidth, υ_r alludes ram capacity.

2) Service-Request Evaluation Cost by Provider: The service provider share the resources based on $Pay-As-You-Go-Model$. It estimates as $t_{ec} = \sum_{dc\in dc}^{n} P_c \times t_e$. Here, t_e is a predicted service request finish time, which remains assigned to ES_j. The service provider has an adaptive asset provision scheme that functions with CPU usage/hour. The provider cost evaluates by considering the service-request finish time of each ES. The outcome multiplied with the cost price of each CPU P_c.

3) Service-Request Evaluation Cost by ES: The SET described as the rate of each service size and ES computing potentiality; it considers the ES RAM, bandwidth capacity. It evaluate as $t_e = 0.0027 \times t_{\mathrm{tt}}$. Based on cost estimation phenomena, the service provider estimates each ES cost as $\mathrm{C}(P) = \sum_{i=1}^{t_i} t_{\mathrm{ec}}^{i}$.

4) Service-Request Queuing Model: The queue length computation method streamlines the SO process by controlling the service-request queue time. It impacts on service-request MS, and retort time; meanwhile, the package policy satisfied service-requests are maintained in the queue set through wait mode. It optimizes the waiting queue length

$$\ell_i^w = \sum_{i=1}^{T} \varpi_i^j \times \left(t_d \Big/ \left(v_{\mathrm{rc}}^j \times v_{\mathrm{mc}}^j \times v_{\mathrm{bc}}^j \right) \right) \tag{18}$$

where ℓ_i^w refers to estimate the capacity of each ES after the execution of all offloaded service-requests

$$v_{\mathrm{rc}}^j = A_{\mathrm{rc}}^j - \sum_{j=1}^{V} \varpi_i^j \times R_{\mathrm{rc}}^j. \tag{19}$$

In (18), the condition of both equations (19) and (20) shows the unavailability of assets to execute the service-request by the respective ES, respectively. In this case, v_{rc}^j, v_{mc}^j esteem do not have positive value, so -10^{-3} multiplied with it's value. We have to add 10^{-3} in case if it is equal to 0

$$v_{\mathrm{mc}}^j = A_{\mathrm{mc}}^j - \sum_{j=1}^{V} \varpi_i^j \times R_{\mathrm{mc}}^j \tag{20}$$

$$v_{\mathrm{bc}}^j = A_{\mathrm{bc}}^j - \sum_{j=1}^{V} \varpi_i^j \times R_{\mathrm{bc}}^j. \tag{21}$$

The queue length value remain increases by multiplying (v_{rc}^j, v_{mc}^j), whenever one of this value became 0 or $-ve$. Therefore, it restricts the service requests to assign without analyzing the resources. It also avoids initiating queues as with wait mode. This step repeats till assigning the service request to the ES, which is recorded for future forthcoming steps.

5) Bounded Threshold Factor: The service-request finish time remains described as a mean rate of service-request capacity and attached ES computing potentiality, and evaluated as $t_c = {}^{t_s}/_{V_{\mathrm{cp}}}$. Here, t_s is the service-request size and V_{cp} is the ES execution capacity based on its resource availability at the current moment. Each ES completion time is evaluates based on the probability model. It considers the current resource status rate and the amount of required resources to execute all assigned service requests. It evaluate as $T_{\mathrm{server}_i} = \sum_{i=1}^{n} t_i^c$. Where n is the total service-requests allocated to ES_j. Total completion time of all ESs are considered to estimate maximum time for fixing upper limit of the system server. It estimates with the following equation:

$$S_{\mathrm{UBL}}^{\mathrm{time}} = \mathrm{Max}\left(T_{\mathrm{server}_1}, T_{\mathrm{server}_2}, T_{\mathrm{server}_3}, \ldots, T_{\mathrm{server}_n}\right). \tag{22}$$

V. DSO Algorithm

Algorithm 1 estimates all preliminary parameters. Line 2, initiates all sets when the PL does not equal to zero. Lines 4–9 have used to determine all service-request S-deadline, original start time, and unique finish time and DAG elements. Lines 11–14 have used to assess the feasible service-request list set based on satisfying the condition about the S-deadline, variance deadline time rate, and it should be less than the approx. If deadline is violated, then the second algorithm became active. Otherwise, the active service-request set updated by service request delete and further not satisfied service-requests forwarded to NFT[t_i] set.

Algorithm 1 PL Enabled Service-Request Sorting Algorithm

input : 1. ES set: $V = \{v_1 + v_2 + v_3 + \ldots\ldots + v_V\}$,
2. SR set $T = \{t_1 + t_2 + t_3 + \ldots\ldots + t_T\}$
3. R: Set of errand entail resources
output: Optimal Number of Feasible SRs

1 Let $T = \{t_1 + t_2 + t_3 + \ldots\ldots + t_T\}$;
2 int T, $FT[t_i] = 0$, $NFT[t_i] = 0$, $WT[t_i] = 0$;
3 **if** $\aleph \neq 0$ **then**
4 **for** *each* $F_i = 1$ *to* F_n **do**
5 Estimation of all PL parameters
 $F_L = \{AT_L, D_L, G_L\}$;
6 **for** *each* $t_i = 1$ *to* T **do**
7 $t_{ij} \in T$;
8 Estimate $d_{i,L}$, $\varepsilon(t_{\mathrm{ic}}^j)_{\mathrm{oft}}$, $\varepsilon(t_{\mathrm{ic}}^j)_{\mathrm{ost}}$;
9 **end**
10 **end**
11 **if** $d_{i,L} \in D_L || d_{i,L} \leqslant D_L || \hat{\partial}_{\mathrm{ic}}^j \leqslant D_L$ **then**
12 Update $FT[t_i] \leftarrow T$;
13 Trigger next algorithm;
14 **end**
15 **else**
16 Estimate $\hat{\partial}_{\mathrm{ic}}^j = d_{i,L} - \varepsilon(t_{\mathrm{ic}}^j)_{\mathrm{ost}}$;
17 *RemoveFun* from SR Set T[t_i];
18 *Update* T[t_i] = T[t_i]-1;
19 $NFT[t_i] \leftarrow T$;
20 **end**
21 **end**
22 Return Feasible Service-requests

In Algorithm 2, lines 2–10 are used to popup the tasks from the pool set since all the Pe errands have been executed without time fluctuation. Line-12 is used to estimate the capacity of the ES. Line-13 is condition $\ell_i^w \geqslant \mathrm{MinBound}\{\ell_i^w\} || MCR\{\varepsilon(t_{\mathrm{ic}}^j)_{\mathrm{ost}}, \varepsilon(t_{\mathrm{ic}}^j)_{\mathrm{lct}}, d_{i,L}\}$ which is used to execute the list of service-requests and also used to execute the waiting requests which have assigned to the ES based on the starting and finishing time rate. Additionally, based on the ES

Algorithm 2 R-Retaliation Algorithm

input : 1. Feasible SR set FT[t_i],
2. ES set: $V = \{v_1 + v_2 + v_3 + \ldots\ldots + v_V\}$,
3. NFT[t_i].
output: Feasible ES set and potential execution of all SRs.

1 int $\ell_i^w \neq 0$;
2 **for** *each* $t_i = 1$ *to* n **do**
3 **for** *each* $t_i \epsilon Su(t_{L,i}^j)$ **do**
4 **if** $Su(t_{L,i}^j) = empty$ **then**
5 Estimate $\varepsilon(t_{\text{ic}}^j)_{\text{oft}}$, $\varepsilon(t_{\text{ic}}^j)_{\text{ost}}$;
6 According outcomes, Update Queue FT[t_i];
7 FT[t_i] $\leftarrow$ NFT[t_i];
8 **end**
9 **end**
10 **end**
11 **for** *each* $V_j = 1$ *to* n **do**
12 Estimate ℓ_i^w;
13 **if**
 $\ell_i^w \geqslant MinBound\{\ell_i^w\}||MCR\{\varepsilon(t_{\text{ic}}^j)_{\text{ost}}, \varepsilon(t_{\text{ic}}^j)_{\text{lct}}, d_{i,L}\}$
 then
14 Estimate $t_{ec} = \sum_{dc \in dc}^{n} P_c \times t_e$,
15 Estimate $t_e = 0.0027 \times t_{tt}$, and $\mathrm{C}(P) = \sum_{i=1}^{t_i} t_{ec}^i$;
16 **if** $v_j^m \geq 0$ *or* $v_j^p \geq 0$ **then**
17 $v_j \leftarrow t_i$;
18 $RemoveFun$ update NF[t_i];
19 Execute all WT[t_i] of ES;
20 **end**
21 **end**
22 **else**
23 Estimate $T_{server_i} = \sum_{i=1}^{n} t_i^c$;
24 $RemoveFun$ update FVT[v_j];
25 Update ES set $\leftarrow v_{j=1}^{V} \leq f(x)$;
26 **end**
27 **end**
28 Return Feasible ESs.

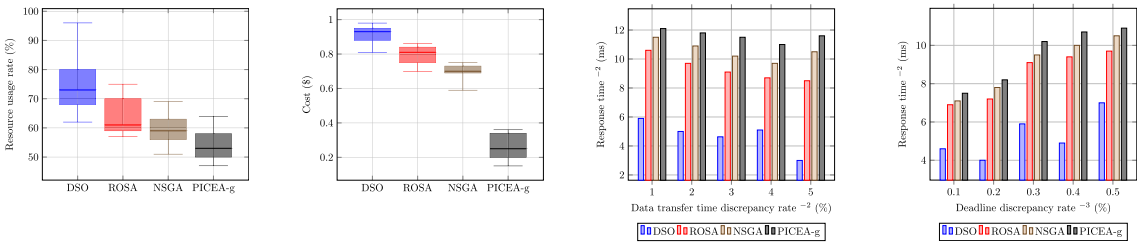
Algorithm 3 SR and Asset Allocation Algorithm

input : 1. FVT[v_j]
2. FT[t_i]
output: Optimal leased cost of ES by Fog

1 Let $FVT[v_j] \neq 0$, $FT[t_i] \neq 0$
2 $\Delta \leftarrow$ Initial unit bill rate;
3 **if** $t_{i\ i\subseteq L}^L \notin F_L$ **then**
4 Assign a service-request to ES;
5 $v_{j+1} \leftarrow t_i^L$;
6 **end**
7 **else**
8 **for** *each* $t_i = 1$ *to* n **do**
9 **for** *each* $v_j = 1$ *to* n **do**
10 **if** v_j *is Idle* **then**
11 $v_j \leftarrow t_i$;
12 Update $\varphi(t_{i,L}^j) \leftarrow$
 $Max\{\hat{\imath}, \hat{\alpha}(t_{ic,L}^j), \max_{t_{i,L}^j \in T_L}\{\alpha(t_{ic,L}^j), w_{i,\hat{i}}^{j,L}\}\}$;
13 **end**
14 **else**
15 Estimate ℓ_i^w;
16 **if** ℓ_i^w *is moderate* **then**
17 $v_j \leftarrow t_i$;
18 Update $\varphi(t_{i,L}^j) \leftarrow$
 $Max\{\hat{\alpha}(t_{ic,L}^j), \max_{t_{i,L}^j \in T_L}\{\alpha(t_{ic,L}^j), w_{i,\hat{i}}^{j,L}\}\}$;
19 **end**
20 **else**
21 Trigger new ES & update active set;
22 Update $WT[t_i] \leftarrow t_i$;
23 **end**
24 **end**
25 Estimate $\hat{\alpha}(t_{ic,L}^j)$ and $\hat{\varphi}(t_{i,L}^j)$;
26 **if** $\hat{\alpha}(t_{ic,L}^j) \leqslant \hat{d}_{i,L}||\hat{\varphi}(t_{i,L}^j) \leqslant \Delta$ **then**
27 Update $\mathbb{Q} \leftarrow FVT[v_j]$;
28 $\Delta \leftarrow \hat{\varphi}(t_{i,L}^j)$;
29 **end**
30 **end**
31 **end**
32 **if** $v_j \notin \mathbb{Q}$ **then**
33 **if** $v_j \vdash \{t_e, \ell_i^w\}$ **then**
34 Select $\begin{pmatrix} v_j \\ \mathbb{Q} \end{pmatrix}$; $v_j \leftarrow t_i$; and Select $\begin{pmatrix} t_j \\ v_j \end{pmatrix}$;
35 **end**
36 **end**
37 **else**
38 **if** $v_j \vdash \hat{\varphi}(t_{i,L}^j)||v_j \vdash \left(\varphi(t_{i,L}^j) \leqslant \hat{d}_{i,L}\right)$ **then**
39 Sort $v_j \leftarrow MinCostSet[v_{j,min}]$;
40 **end**
41 **else**
42 Sort $v_j \leftarrow MaxRankSet[v_{j,mr}]$;
43 **end**
44 **end**
45 **end**

capacity, an arrival service request would be assigned. In case the condition does not satisfy, then the respective ES will be treated as not feasible using (13). The scheduling policy is initiated using Algorithm 3.

Algorithm 3 schedules the service requests in the queue based on ES capacity instances and outcomes of Algorithms 1 and 2, respectively. Line 3 is used to cross-check each service request belongs to the same PL or not. If not: the service request is allocated to a new ES based on its capacity and service-request deadline. If yes: the fundamental steps are carried out to accomplish the objective. Lines 10–13 have been used to assign service-request to ES based on Line 12 when ES is in an idle state. Otherwise, based on ES capacity, the service requests are assigned based on the additional condition as per line 18. Otherwise, the service request is moved to the PL pool or assigned to another ES. Lines 25–28 have been used to update the feasible ES set based on the task completion (TC) with a time variance rate. In case, if ES does not belong to the ES set, then ESs are ordered based on line 33. Line 40 shows the procedure to select ES from the low leased cost ES set; otherwise, have to choose from a high-ranked ES set to accomplish the objective.



(a) (b) (c) (d)

Fig. 4. Impact of deadline discrepancy on cost and asset usage. (a) RUR analysis. (b) Impact of cost analysis and Average SET analysis of proposed and existing systems. (c) Impact of data transfer time discrepancy rate. (d) Deadline discrepancy time impact analysis on response time.

VI. Experimental Results

A. Performance Metrics

This section dives into all metrics to examine the proposed algorithm's accuracy compared to the benchmark existing systems.

1) *Average Response Time:* It estimates ESs Ψ_i^j response time of fog $\Psi(h)$ as per the capacity and assigned workload. It assess with the following equation:

$$\Psi_i^j = \mathrm{Min}\left(\frac{\sum\left(t_i^{\mathrm{ost},j} + t_i^{\mathrm{oft},j}\right)}{\mathrm{B}_j}\right) \quad \text{and} \quad \Psi(h) = \sum_{j=1}^{\mathrm{ES}_j} \Psi_j^h. \tag{23}$$

2) *Asset Usage:* It uses to calculate usage rate of the ES ξ_j^h on the fog node $\xi(h)$ based total active B_j and overall processing time A_j in the period. It assess with the following equation:

$$\xi_j^h = \frac{\sum_{j=1}^{|\mathrm{ES}_j|} \mathrm{A}_j}{\sum_{j=1}^{|\mathrm{ES}_j|} \mathrm{B}_j} \quad \text{and} \quad \xi(h) = \sum_{j=1}^{|\mathrm{ES}_j|} \xi_j^h. \tag{24}$$

3) *Deadline Violation Rate* Λ*:* It uses to calculate the average length deadline violation rate of each PL D_V from PL set $\aleph$. It assess with the following equation:

$$D_V = \sum_{L=1}^{\aleph} \frac{\aleph_{\mathrm{Pr}}}{\aleph}, \; \aleph_{\mathrm{Pr}} = \left(\mathrm{MS}_L + \mathrm{AT}_L - D_L\right)/D_L - \mathrm{AT}_L \tag{25}$$

where $\hat{\aleph}$ refers to a number of violated PLs among all PLs $\aleph$. Therefore $\Lambda = \hat{\aleph}/\aleph \times 100$.

B. Simulation Details

To assess the accuracy of our proposed framework, we have utilized the FogSim toolbox, PC with i5 processor Intel center, 16 GB RAM, and 2.40 GHz CPU limit with a versatile hard disk. The proposed approach designed a `DatacenterBroker`, in which the service-request distribution is carried out according to the computational workload mechanism. The simulation has been iterated a few times, for example, the simulation conducted 25 times, and the parameter outcomes are recorded in Table I. The proposed approach is simulated with a real-time workload data such as Laser Interferometer Gravitational-Wave Observatory (`LIGO`), `Montage` and `Cybershake`. The workloads with various sizes such as 100, 150, and 250 errands with the DAG phenomenon.

TABLE I
Fog Instances

ESType	CPU	RAM	Disk	$/hr	ComT(ms)
m2.6xlarge	9	8GB	64GB	0.95-0.25	3.50
m2.2xlarge	8	8GB	32GB	0.45-0.18	3.21
m1.2xmedi	6	6GB	32GB	0.24-0.19	4.16
m1.5xmedi	6	8GB	64GB	0.21-0.15	4.92
m1.xsmall	4	6GB	32GB	0.13-0.17	6.45
m1.xsmall	6	6GB	32GB	0.15-0.87	6.12

TABLE II
Various PLs

Name	Label	Input size (GB)	Deadline (Min)
LIGO	$i_1/i_2/i_3/i_4$	50/40/36/14	85/49/30/24
Montage	$i_5/i_6/i_7/i_8$	45/32/21/11	51/43/17/9
Cybershake	$i_9/i_{10}/i_{11}$	80/78/30	75/48/43

3-Process workload (PL) of application: 4 *different sizes:* 100 *various PL*, which differs in computation entailment's and weight of edges and have listed in Table II. Total 1200 PL has been taken into consideration to test the performance of our proposed system. Also, the proposed algorithm has been examined in a heterogeneous environment, and the results are contrasted with standard benchmark state-of-art approaches on Elastic Amazon Web Services (AWS) IaaS provider to analyze unit leased cost.

Fig. 4 shows an accurate usage efficiency and average SET analysis of DSO assets compared to three existing systems. DSO regulates initial time attributes to avoid service-requests execution delay. In the second step, DSO identifies suitable machines to accommodate service requests for execution; it gives an impact on ES idle time and low asset usage utilization. Average finish time is the benchmark to test the time efficiency of our proposed and the existing systems.

Subsequently, Fig. 4 also illustrates the response time consistency of DSO compared to existing systems. The DSO approach has achieved low response time (high performance) than the remaining strategies because of S-deadline estimation. Fig. 4(c) illustrates the response time of the proposed system, which is better at the worst time discrepancy rate than other approaches. Fig. 4(d) shows the same trend during the deadline discrepancy rate, but the PICES-g releases a high number of ES because of lack in asset usage, which confers a more waiting time of service-request, and strives for suitable ES.

Fig. 5(a) illustrates asset usage efficiency at all extreme levels because of a more significant amount of service requests. Fig. 5(b) shows the individual errands deadline violation count of DSO, which is comparatively low than the other three approaches over adaptive PL allocation. Fig. 5(c) shows each machine leased cost, and it is moderately too small than

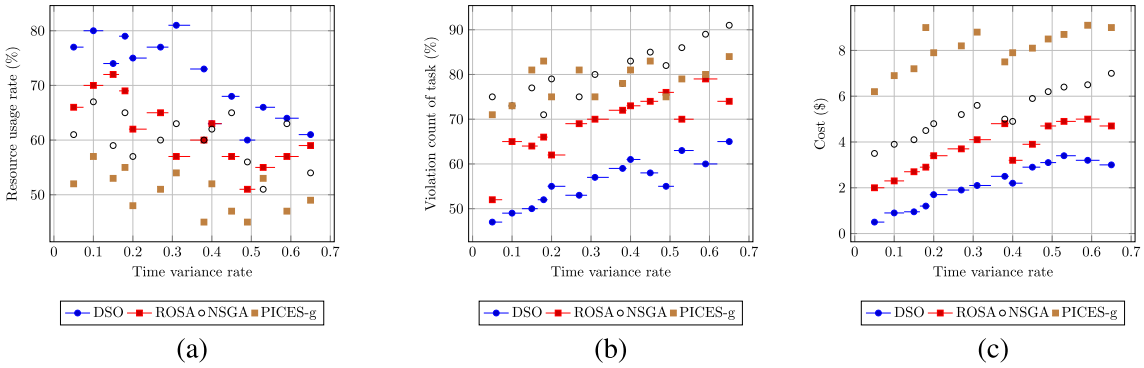


Fig. 5. Analysis of service-request execution time variance rate. (a) Impact of RUR. (b) Deadline discrepancy time impact analysis. (c) Cost analysis.

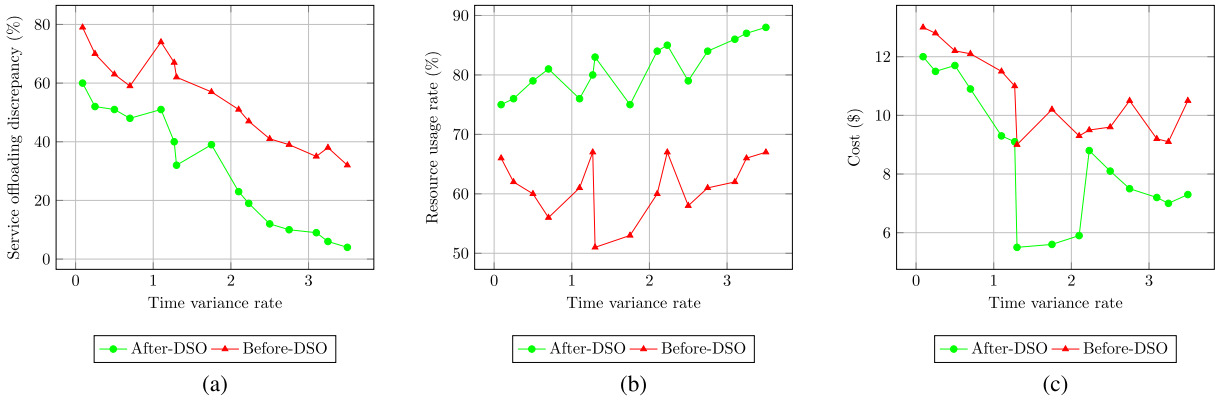


Fig. 6. Average execution time analyses of the proposed and existing systems. (a) Impact of migration time discrepancy. (b) Impact of RUR. (c) Impact of processing cost.

TABLE III
OBJECTIVE RESULTS OF TEST EXPERIMENTS

Algorithms	Average response time		Total cost ($)		SLA violation rate (%)	
	16ES	32ES	16ES	32ES	16ES	32ES
DSO	171.56	77.41	89.51	105.54	51.49	54.61
ROSA	195.01	95.31	106.20	109.61	61.89	57.05
NSGA	215.32	96.52	108.98	112.15	61.95	58.78
PICES-g	216.05	97.85	111.41	115.52	62.79	59.25

different methods; because our proposed system appraises all entail parameters before assigning the service request to a suitable ES. Therefore, it makes a difference in billing leased costs. Fig. 6(a), alludes individual service-request violation rate. Here, the average SET is 1.5 and consider it as a benchmark to analyze the service-request violation rate before and after the DSO approach.

Fig. 6(b) illustrates assets usage analysis reports after and before DSO. We noticed that the asset usage rate has been increased tremendously after the DSO approach than before. Fig. 6(c), illustrates that the leased cost is less after the R-retaliation process since the process dramatically reduces fluctuations, and asset wastage even at service-request execution rate is equal to zero. After the 1.5, the leased cost has been diminished drastically.

Table III illustrates comparative outcome analysis among extant approaches in terms of leased cost, average response time, SLA violation rate with two scenarios, such as 16 ES and 32 ES. The outcomes show that in all three aspects, our proposed system has an accurate performance. The experiments have illustrated that R-retaliation impacts on DSO approach outcomes more than extant algorithms due to prior estimation of service execution time, and respective parameters.

VII. CONCLUSION

The designed two-step DSO approach has achieved strong service reliability on the considered SIoT application workload to meet the requirements of edge infrastructure and Industry 4.0 services. The DSO approach has optimized the haphazard service allocations based on `bindServiceToServer(·)` strategy using `Datacenter-Brocker`. The workload analyzer estimates the weight of PL to share the resources with 87% accuracy based on S-deadline and prognosticated completion time factors. The ES capacity weight factor selects a suitable server to avoid execution hiccups, which reduces the deadline violation rate of individual services and the cost of a server. The R-retaliation model evaluates the most favorable resource and service allocation policy to balance the tradeoff between the execution of SO data, and workload deadline violation rates. Our approach has achieved 85% QoS as per service arrival rate based on the R-retaliation method. Our method, in turn, offers service providers to upgrade the selected target workload objectives to reduce the cost by at least 46%, deadline violation rate by at least 79% than state-of-art approaches, even at high workloads. In the future, design a quantum integrated edge orchestration to achieve optimized SO with high service reliability.

APPENDIX

Time complexity of the proposed algorithms are essential to streamline the objective.

Theorem A1: $O(|\Lambda|^2+|\Delta|\log(|\Delta|)+|\Delta|^2\times|\Theta|)$ is the time complexity of our initial algorithm, i.e.; Algorithm 1.

Proof: DAG graph (G_L) enables 4-constructive parameters $G_L = \{T_L, E_L, \Gamma_L, W_L\}$, where $T_L = \{t_{1,L}, t_{2,L}, t_{3,L}, \ldots, t_{n,L}\}$ number of tasks, E_L refers edges between two service tasks and is evaluated with the following equation:

$$|E_L| = \frac{|\Lambda| \times (|\Lambda| - 2)}{2}. \tag{A.1}$$

The estimation of this parameters $\varepsilon(t_{\text{ic}}^j)_{\text{ost}}$, $\varepsilon(t_{\text{ic}}^j)_{\text{lct}}$, $\wp(t_{\text{ic}}^j)$ might required some time, which is labeled with $O(|\Lambda|^2)$. Individual evaluation of service requests deadline would also required some time, which is labeled with $O(|\Lambda|)$.

Hence, our initial approach optimal complexity of time is $O(|\Lambda|^2)$; by clubbing of $O(|\Lambda|^2)$ and $O(|\Lambda|)$. Additionally, the length of the queue is estimated with the following equation:

$$L(Q) = |\Lambda|. \tag{A.2}$$

The process of service-requests sorting would require an ample of time, which is denoted with $O(|\Delta|\log(|\Delta|))$.

The server active time and their maintenance require $O(|\Delta| \times |\Theta|)$. Because, the $\Pr(t_{L,i}^j) = V_1^L$ requires an ample of time $O(|\Delta|)$ to streamline the objective

$$\begin{aligned}
&= O(|\Lambda|) \quad \therefore L(Q) = |\Lambda| \\
&= O\big(|\Lambda|^2\big) \\
&= O(|\Delta|\log(|\Delta|)) \\
&= |\Delta| \times (O(|\Delta| \times |\Theta|) + O(\log(|\Delta|)) + O(|\Delta|)) \\
&= O(|\Lambda|^2 + |\Delta|\log(|\Delta|) + |\Delta|^2 \times |\Theta|).
\end{aligned}$$

As we stated earlier, the time complexity of the initial level includes service length queue which helps to offload the services to the suitable server to achieve low execution time and complexity, which is defined with the following equation:

$$O(|\Lambda|^2 + |\Delta|\log(|\Delta|) + |\Delta|^2 \times |\Theta|). \tag{A.3}$$

Theorem A2: $O(|\nabla|\log(|\nabla|) + |\Delta||\nabla| \times |\Theta|)$ is the time complexity of our R-retaliation algorithm, i.e.; Algorithm 2.

Proof: Algorithm 2 primarily estimates the pre and post service completion time, i.e., $\Pr(t_{L,i}^j) = V_1^L$ $\text{Su}(t_{L,i}^j) = V_7^L$ and it entails time complexity. It is estimated with the following equation:

$$O(|\nabla|). \tag{A2.1}$$

But repeatedly have to ensure about completion of $\Pr(t_{L,i}^j) = V_1^L$ pre service completion status. Therefore, the complexity of this process is the following equation:

$$O(|\Delta|). \tag{A2.2}$$

In such case, might have to do few adjustments, so the complexity of this process is (A2.3) and it is carried out on the respective server and the time complexity is the following equation:

$$O(|\Delta||\nabla|) \tag{A2.3}$$

$$O(|\Theta|)$$

$$O(|\Delta||\nabla| \times |\Theta|). \tag{A2.4}$$

Therefore, an optimal complexity remain accomplished by adding these two process complexities

$$\begin{aligned}
&= O(|\Delta||\nabla|) \\
&= O(|\Delta||\nabla| \times |\Theta|) \\
&= O(|\nabla|\log(|\nabla|)) + O(|\Delta||\nabla| \times |\Theta|) \\
&= O(|\nabla|\log(|\nabla|) + |\Delta||\nabla| \times |\Theta|).
\end{aligned}$$

REFERENCES

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

[2] C. Jiang, X. Cheng, H. Gao, X. Zhou, and J. Wan, "Toward computation offloading in edge computing: A survey," *IEEE Access*, vol. 7, pp. 131543–131558, 2019.

[3] G. S. Aujla, N. Kumar, A. Y. Zomaya, and R. Ranjan, "Optimal decision making for big data processing at edge-cloud environment: An SDN perspective," *IEEE Trans. Ind. Informat.*, vol. 14, no. 2, pp. 778–789, Feb. 2018.

[4] X. Xu, Q. Wu, L. Qi, W. Dou, S.-B. Tsai, and M. Z. A. Bhuiyan, "Trust-aware service offloading for video surveillance in edge computing enabled internet of vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 3, pp. 1787–1796, Mar. 2021.

[5] H. Wang, Y. Wu, G. Min, J. Xu, and P. Tang, "Data-driven dynamic resource scheduling for network slicing: A deep reinforcement learning approach," *Inf. Sci.*, vol. 498, pp. 106–116, Sep. 2019.

[6] M. Mekala *et al.*, "Deep learning-influenced joint vehicle-to-infrastructure and vehicle-to-vehicle communication approach for internet of vehicles," *Expert Syst.*, p. e12815, Oct. 2021. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/exsy.12815

[7] X. Xu, Q. Huang, X. Yin, M. Abbasi, M. R. Khosravi, and L. Qi, "Intelligent offloading for collaborative smart city services in edge computing," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 7919–7927, Sep. 2020.

[8] Y. Cheng, "Modeling of manufacturing service supply–demand matching hypernetwork in service-oriented manufacturing systems," *Robot. Comput.-Integr. Manuf.*, vol. 45, pp. 59–72, Jun. 2017.

[9] S. Song, S. Ma, J. Zhao, F. Yang, and L. Zhai, "Cost-efficient multi-service task offloading scheduling for mobile edge computing," *Appl. Intell.*, vol. 52, pp. 4028–4040, Jul. 2021.

[10] R. Zhang, K. Wu, M. Li, and J. Wang, "Online resource scheduling under concave pricing for cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 4, pp. 1131–1145, Apr. 2016.

[11] X. Chen and D. Long, "Task scheduling of cloud computing using integrated particle swarm algorithm and ant colony algorithm," *Cluster Comput.*, vol. 22, no. S2, pp. 2761–2769, Mar. 2019.

[12] K. Baital and A. Chakrabarti, "Dynamic scheduling of real-time tasks in heterogeneous multicore systems," *IEEE Embedded Syst. Lett.*, vol. 11, no. 1, pp. 29–32, Mar. 2019.

[13] D. Zeng, L. Gu, S. Pan, J. Cai, and S. Guo, "Resource management at the network edge: A deep reinforcement learning approach," *IEEE Netw.*, vol. 33, no. 3, pp. 26–33, May 2019.

[14] P. Paknejad, R. Khorsand, and M. Ramezanpour, "Chaotic improved PICEA-G-based multi-objective optimization for workflow scheduling in cloud environment," *Future Gener. Comput. Syst.*, vol. 117, pp. 12–28, Apr. 2021.

[15] M. S. Mekala and P. Viswanathan, "Energy-efficient virtual machine selection based on resource ranking and utilization factor approach in cloud computing for IoT," *Comput. Elect. Eng.*, vol. 73, pp. 227–244, Jan. 2019.

[16] L. Chiaraviglio, F. D'Andreagiovanni, R. Lancellotti, M. Shojafar, N. Blefari-Melazzi, and C. Canali, "An approach to balance maintenance costs and electricity consumption in cloud data centers," *IEEE Trans. Sustain. Comput.*, vol. 3, no. 4, pp. 274–288, May 2018.

[17] M. S. Mekala, A. Jolfaei, G. Srivastava, X. Zheng, A. Anvari-Moghaddam, and P. Viswanathan, "Resource offload consolidation based on deep-reinforcement learning approach in cyber-physical systems," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 6, no. 2, pp. 245–254, Apr. 2022.

[18] Z.-G. Chen *et al.*, "Multiobjective cloud workflow scheduling: A multiple populations ant colony system approach," *IEEE Trans. Cybern.*, vol. 49, no. 8, pp. 2912–2926, Aug. 2019.

[19] M. S. Mekala, R. Patan, S. H. Islam, D. Samanta, G. A. Mallah, and S. A. Chaudhry, "DAWM: Cost-aware asset claim analysis approach on big data analytic computation model for cloud data centre," *Secur. Commun. Netw.*, vol. 2021, pp. 1–16, May 2021.