Journal of Theoretical and Applied Information Technology
15th September 2022. Vol.100. No 17

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

OPTIMAL OMNIDIRECTIONAL OFFLOADING IN FEDERATED CLOUD-FOG SYSTEMS WITH COST MINIMIZATION

SAUMYARANJAN DASH[1], RAJ KUMAR PARIDA[2], JYOTIPRAKASH DASH [3], ASIF UDDIN KHAN[4], SANTOSH KUMAR SWAIN[5]

[1, 5] School of Computer Engineering, KIIT Deemed to be University, Bhubaneswar, India
[2] Department of Computer Science & Engineering, Koneru Lakshmaiah Education Foundation (KLEF), Vaddeswaram, Andhra Pradesh, India.
[3] Department of Computer Science and Engineering, Gandhi Institute For Technology, Bhubaneswar, India
[4] Department of Computer Science and Engineering, Silicon Institute of Technology, Bhubaneswar, India
E-mail: [1] soumya@silicon.ac.in, [2] rajkumarparida@gmail.com, [3] jyotiprakash.dash@gift.edu.in, [4]asif.khan@silicon.ac.in, [5] sswainfcs@kiit.ac.in

ABSTRACT

Cloud computing technology has brought a significant impact on the field of technology. Similarly, fog computing provides similar facilities with a lesser coverage area but is closer to the user. In a federated environment, these technologies can complement each other by offloading their request to improve the services. In this paper, we propose a federated two-tier architecture consisting of cloud and fog, where the cloud has a higher cost and fog has a lower cost. In this architecture, omnidirectional offloading is considered in which offloading can be done vertically and horizontally, such as Fog to Cloud, Cloud to Fog, and Fog to Fog. We propose a capacity optimization problem in this federated architecture to optimize the capacities to minimize the total cost. We are using a modified simulated annealing algorithm to solve this optimization problem. The results show the performance of our algorithm is nearly optimal and better than the SA in terms of execution time, and our proposed architecture can save more cost than other existing architectures.

Keywords: *Cloud, Fog, Federation, Offloading, Cost, Capacity*

1. INTRODUCTION

Cloud Computing facilitates computing, storage, and network resources for the users on demand through the network, reducing the overall cost of the user and enabling profit maximization of the service provider [1]. But the service delay in cloud computing is not acceptable for numerous numbers of delay-sensitive real-time applications where the task must be completed within a certain deadline. Since cloud servers are far away from the end-users, it is not possible to guarantee QoS and QoE for real-time applications at the edge of the network [2]. Further, with the advancement in computing, communication, and network technologies, users want to use the computing services anywhere, anytime, and through any device. To address the issues of cloud computing, such as delay and bandwidth consumption, a new computing paradigm called Fog Computing was introduced, as shown in Fig. 1, where cloud resources are provided at the edge of the network [3]. Unlike cloud computing, fog computing is intended for distributed computing, where numerous peripheral devices with cloud like properties are closer to the users. The idea is to do as much processing as possible using computing units closer to the user rather than forwarded to the cloud, which requires bandwidth and latency [4]. Due to the tremendous growth in computing services and devices, processing requests at the fog are also increased. As fog devices are resource constrained, the requests are offloaded from the devices to other fog nodes and cloud server, if they are not able to be processed locally [5], [6], [7].

Various research has already been carried out on offloading from edge to fog and from fog to cloud, where offloading is done only in a unidirectional manner. In this paper, to further optimize the latency and cost, we propose a federated two-tier cloud-fog omnidirectional architecture consisting of cloud and fog, where the cloud has a higher cost and fog has a lower cost. In this context, "federation" means the collection of clouds cooperating to provide resources that are requested by users [8]. One can offload to another whenever required to provide better services to respective subscribers. In a federated cloud-fog system, the cloud and fog servers cooperate with each other to process their tasks by offloading service requests to each other. The federation has

various advantages, such as load balancing, dynamic resource scaling, and benefits in terms of economic perspective.

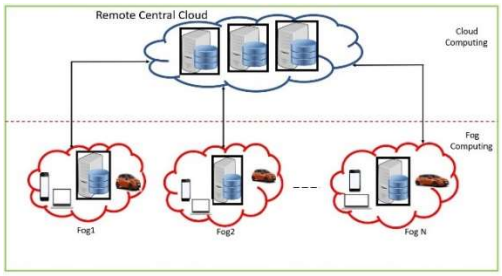

Figure 1: Cloud-fog systems

This paper proposes a capacity optimization problem in the cloud-fog federated system where the offloading scenario is omnidirectional.

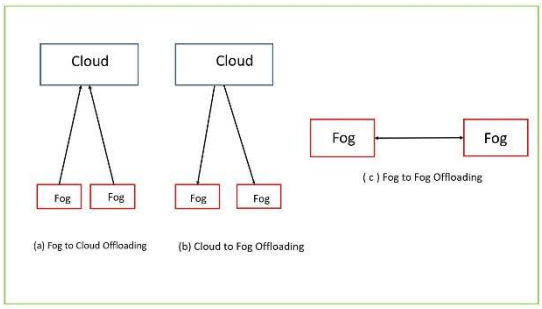

Figure 2: Omnidirectional cloud-fog offloading.

In our proposed federated architecture, we have used Omnidirectional offloading in which offloading can be done both vertically and horizontally, such as fog to cloud, cloud to fog, and fog to fog offloading, as shown in Fig. 2(a), 2(b), and 2(c), respectively. We propose a modified simulated annealing algorithm to solve this optimization problem to minimize the total cost, which includes the computation and communication costs with latency as a constraint. The unique contribution of this paper is as follows.

1) We proposed a novel cloud–fog federated architecture with Omnidirectional offloading.
2) We proposed a capacity optimization problem in the proposed federated architecture to minimize the cost where latency is the constraint.
3) Then we proposed a modified Simulated Annealing algorithm to solve this optimization problem, reduce execution time and save more cost.

The remaining part of the paper is organized as follows. In Section 2, we present the research works related to offloading in fog computing environment and in cloud- fog federation. The proposed two-tier federated cloud fog architecture, the system model and the problem formulation are illustrated in Section 3. We put forward the detailed algorithm design and solution approach in Section 4. In Section 5, we discuss the performance analysis and simulation results. Finally, the paper is concluded in Section 6.

2. RELATED WORK

In this section, we present various works on offloading in different scenarios. [16], [25], and [30] present the offloading schemes in edge computing. [9], [11], [15], [17], [18], [29], and [31] present the offloading schemes in fog computing. [23], and [24] discuss the cloud federations. [21], and [27] illustrate the offloading schemes in mobile edge computing. [12], [20], and [26] describe the offloading schemes in edge-cloud systems. [13], [14], [19], [32], and [34] discuss the offloading schemes in fog-cloud systems. [10] presents an offloading mechanism in vehicular fog and cloud system. [35], and [37] present the offloading mechanisms in federated cloud-edge systems. [22], and [33] describe the offloading schemes in federated cloud-edge-fog.

Lin et al. investigated edge computing offloading modeling by considering different edge computing architectures [16]. Cao et al. presented an integrated model for service provisioning between edge infrastructure providers by adopting a cost-efficient resource management scheme [25]. Bi et al. proposed a heterogeneous edge offloading scheme to minimize the total consumption of energy [30].

Zaharia et al. presented a complete machine-learning-based solution for offloading of traffic in the fog networks. [9]. Wei et al. proposed a downlink scheme for computation offloading in a fog environment consisting of multiple fog nodes and an IoT device with the aim of maximizing the system utility and meeting the energy and delay constraints [11]. Liu et al. proposed a dynamic game-theory-based socially-aware computation offloading method to minimize the execution cost [15]. Jiang et al. presented a fog computing-based energy-efficient offloading mechanism by effective management of communication and computation resources to meet the response time [17]. Jiang et al. presented an offloading scheme in a shared fog network by proposing a non-linear integer model for efficient task scheduling [18]. The optimal offloading decision for achieving a better quality of experience (QoE) is perceived as a key research problem in [29] considering the inherent characteristic of tasks.

Aazam et al. in [31] discussed the recent schemes for offloading in the domains of cloud, fog, and IoT. The authors also explained the key factors important to consider offloading in each scenario.

Mashayekhy et al. presented a game theory-based model in which business structure is reshaped among different cloud providers. A cloud federation mechanism was proposed by the authors where the cloud providers' profit is maximized by reduced resource utilization [23]. Chen et al. described the integration of horizontal and vertical cloud federation [24].

Arif et al. proposed an LSTM-based offloading architecture and presented a routing scheme for an efficient computational offloading strategy to reduce the delay and energy consumption and increase the security level of the devices [21]. Zhao et al. presented a MEC-Cloud collaborative offloading approach to optimize the allocation of computational resources and offloading decisions [27].

Wang et al. presented a survey article that represents various challenges, techniques, and state-of-the-art regarding the offloading problem. The authors also briefly compared the performance of several offloading algorithms for edge-cloud-based scenarios [12]. Wang et al. presented a survey article on cooperative edge-cloud task offloading [20]. Thai et al. proposed a computing architecture for cloud edge that provides both horizontal and vertical offloading in order to optimize the capacity and workload to minimize the total cost [26].

Mansouri et al. presented a fog-cloud hierarchical computation offloading paradigm to improve the QoE and reduce the delay and energy [13]. Meng et al. presented a hybrid computation offloading technique by taking into account the different capabilities of fog and cloud servers to optimize the energy consumption while meeting the delay constraint [14]. Du et al. addressed mixed fog-cloud offloading by optimizing resource allocation, bandwidth, and transmit power while satisfying the maximum delay [19]. Mahini et al. proposed architecture and devised an evolutionary model to address task offloading in an IoT environment [32]. Besharati et al. proposed an auction mechanism-based fog-cloud offloading method by considering the physical infrastructure's limitations and specifications [34].

Wu et al. proposed a scheme for task offloading by considering the departure of occupied vehicles [10].

Kar et al. presented an omnidirectional offloading architecture for a federated cloud-edge system with the objective of optimizing dual cost while considering latency constraints [35]. Akutsu et al. proposed two models namely VDV and VDHS for offloading based on queue-length thresholds at user equipment and edges [37].

A comprehensive description of different offloading schemes used for various federated scenarios and also the current research status was discussed in [22]. This paper also presented a comparative study of traditional optimization and machine learning approaches on various federated architectures. Hwang et al. presented a delay constraint-based offloading optimization strategy for a federated three-tier environment [33].

From the literature, we observed that various research has already been carried out on offloading from edge to fog, edge to cloud, and from fog to cloud, where offloading is done only in a unidirectional manner. In this paper, we propose a federated two-tier cloud-fog omnidirectional architecture in which offloading can be done both vertically and horizontally, such as fog to the cloud, cloud to fog, and fog to fog offloading.

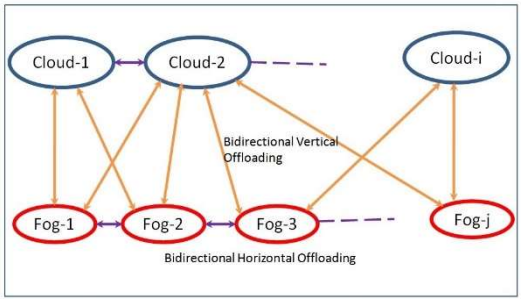

Figure 3: Two-tier federated cloud-fog architecture

3. SYSTEM MODEL AND PROBLEM FORMULATION

3.1 Two Tier Federated Cloud Fog Architecture

Federated architecture is a collection of servers of different service providers, where the service providers co-operate with each other and share their resources. One service provider can rent its resources to other service providers. In some scenarios, a service provider can act as a service provider and a user simultaneously. In this architecture, the user's request for a service provider can be processed by another service provider by offloading the requests. The federated architecture is very useful and economical in the cloud computing environment, where the nature of the user's demand is very dynamic. The federated architecture is managed by an entity known as Federation Manager, which is the agent responsible for the federation agreement between two service providers. There are two types of the federation, namely, horizontal

Journal of Theoretical and Applied Information Technology
15th September 2022. Vol.100. No 17

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

federation and vertical federation. The service providers are on the same level in a horizontal federation, and in a vertical federation, they are on different levels.

In our proposed federated architecture, we have used the cloud servers and fog servers as the federating nodes on two levels. Clouds are in the upper level and fogs are in the lower level as shown in Fig. 3. Offloading can be done horizontally between cloud to cloud and fog to fog in a bidirectional manner, similarly vertical offloading can be done between cloud to fog and fog to cloud in the bidirectional manner as shown in Fig. 3.

As per our proposed architecture, we consider a two-tier cloud-fog system, which consists of multiple clouds and fogs, as illustrated in Fig. 4. All fogs and clouds are connected vertically, and all fogs are connected horizontally. Here, not only fog can offload to the cloud, but the cloud can also offload to the fogs [22]. And fogs can offload to each other horizontally. Horizontal offloading is the scenario, where the request for fog is served by another fog. Vertical downward offloading is when the request for a cloud is served by the fog [26], while vertical upward offloading is when a request for the fog is served by the cloud [25].

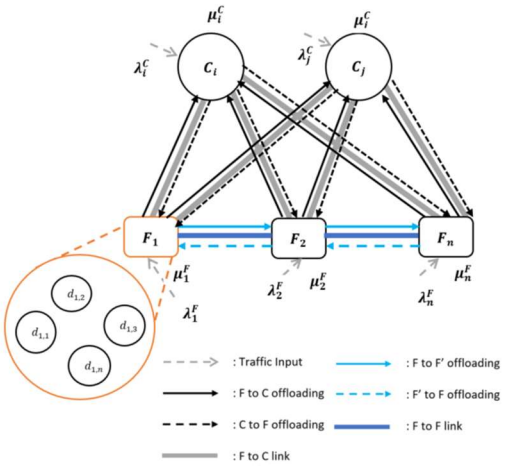

Figure 4: Offloading in Two-tier federated cloud-fog architecture.

3.2 System Model

In this paper, the cloud is denoted by C_i, $i = \{1, 2, \ldots, m\}$, and m is the total number of clouds in the system. The fog is denoted by F_j, $j = \{1, 2, \ldots, n\}$, where, n is the total number of fogs in the system. The arrival request coming to cloud node C_i with capacity μ_i^C is denoted as λ_i^C. The arrival request coming to fog F_j capacity μ_j^F is denoted as λ_j^F. As mentioned earlier, our architecture considered horizontal offloading between the fog, which means there will be channel bandwidth for communication between fogs denoted by $B_{Fj,Fj'}$ and offloading task between two fogs denoted as $O_{Fj,Fj'}$. The vertical offloading in this architecture consists of vertical upward offloading and vertical downward offloading. For the vertical offloading between cloud and fog, the channel bandwidth is denoted by $B_{Ci,Fj}$. For the vertical downward offloading, where a request is offloaded from the cloud to the fog, denoted by $O_{Ci,Fj}$, and for the vertical upward offloading, where requests are offloaded from the fog to the cloud, denoted by $O_{Fj,Ci}$. Each fog consists of several devices denoted as $d_{j,k}$. All the variables and notations used in this paper are presented in Table I.

We assume requests coming to the cloud and fog will be Poisson processes with an average arrival rate of λ_i^C and λ_j^F. And for the traffic model at the cloud and fog in this research, we are considering the M/M/1 queue. Every request which comes either to the cloud or fog will need time to be processed which is the time to compute request or computation for latency can be denoted as L_{C_i} that can be expressed as

$$L_{C_i} = \frac{1}{\mu_{C_i}^{MAX} - (\lambda_i^C + \sum_{j=1}^{n} O_{F_j,C_i} - \sum_{j=1}^{n} O_{C_i F_j})} \quad (1)$$

The cloud computing capacity is usually larger than the fog computing capacity, which explains computing latency in the cloud is faster than in fog. In fog, the time that is consumed to process the request in fog can be denoted as L_{F_j} that can be expressed as

$$L_{F_j} = \frac{1}{\mu_{F_j}^{MAX} - (\lambda_j^F + \sum_{i=1}^{m} O_{C_i,F_j} + \sum_{j'=1}^{n} O_{F_{j'},F_j} - \sum_{j'=1}^{n} O_{F_j F_{j'}} - \sum_{i=1}^{m} O_{F_j C_i})} \quad (2)$$

Either cloud or fog, when it can't execute or handle requests efficiently, it will try to offload either horizontally or vertically. As we know cloud computing cost is much higher than the fog as a result requests are offloaded from cloud to fog. As we offload a request, communication latency is considered as a key performance indicator. Communication latency from cloud to fog can be denoted as L_{C_i,F_j} and expressed as

$$L_{C_i,F_j} = \frac{1}{B_{C_i,F_j} - O_{C_i,F_j}} \quad (3)$$

Table I: Variables and Notations

Notation	Description
System:	
C_i, $1 \le i \le m$	i^{th} cloud
F_j, $1 \le j \le n$	j^{th} fog
$d_{j,k}$, $1 \le j \le n$, $1 \le k \le n_j$	k^{th} device in j^{th} fog
Traffic:	
λ_i^C, λ_j^F	traffic coming into the i^{th} cloud and j^{th} fog
Capacity:	
B_{C_i,F_j}, $B_{F_j,F_{j'}}$	Communication capacity between C_i, F_j and F_j, $F_{j'}$
$\mu_{C_i}^U, \mu_{C_i}^E, \mu_{C_i}^{MAX}$	In use, exceeding and maximum computing capacity of i^{th} cloud
$\mu_{F_j}^U, \mu_{F_j}^E, \mu_{F_j}^{MAX}$	In use, exceeding and maximum computing capacity of j^{th} fog
Latency:	
W_{C_i}, W_{F_j}	Waiting time in the i^{th} cloud and j^{th} fog
L_{C_i}, L_{F_j}	Computing latency at C_i and F_j
$L_{c_i,F_j}, L_{F_jC_i}, L_{F_jF_{j'}}$	Communicating latency from C_i to F_j, from F_j to C_i, and from F_j to $F_{j'}$
L_{MAX}	Maximum latenc
Cost:	
τ_{total}, τ_{comm}, τ_{comp}, τ_{wait}	Total cost, communication cost, computation cost, and waiting cost
τ_C, τ_F	Total computing cost in cloud and fog
τw_c, τw_F	Total waiting cost in cloud and fog
$\tau^{F\to F}$, $\tau^{C\to F}$, $\tau^{F\to C}$	Total communication cost from fog to fog, from cloud to fog, and from fog to cloud
φ_{C_i}, φ_{Fj}	Unit computing cost of C_i and F_j
wC_i,F_j, $wF_j,F_{j'}$	Unit communication cost between C_i and F_j, between F_j and $F_{j'}$
S_{C_i}, S_{F_j}	Unit waiting cost for i^{th} cloud and j^{th} fog

The communication latency, when requests are offloaded from fog to cloud denoted as L_{F_j,C_i} can expressed as

$$L_{F_j,C_i} = \frac{1}{B_{C_i,F_j} - O_{F_j,C_i}} \quad (4)$$

The fog to fog communication latency denoted as $L_{F_j,F_{j'}}$ can be calculated as

$$L_{F_j,F_{j'}} = \frac{1}{B_{F_j,F_{j'}} - O_{F_j,F_{j'}}} \quad (5)$$

In general, while using the cloud and fog, the resources such as CPU and storage are considered as entities that incurs cost. The computing cost in the cloud is usually much higher than fog [7]. Total computing cost τ_C for the clouds can be derived as

$$\tau_C = \sum_{i=1}^{m} \left(\mu_{C_i}^U \times \varphi_{C_i}\right) \quad (6)$$

Then, the total computing cost for fogs denoted as τ_F can be expressed as

$$T_F = \sum_{j=1}^{n} \left(\mu_{F_j}^U \times \varphi_{F_j}\right) \quad (7)$$

The total communication cost of cloud to fog offloading denoted as $\tau^{C\to F}$ and can be calculated as

$$\tau^{C\to F} = \sum_{i=1}^{m} \sum_{j=1}^{n} \left(O_{C_i,F_j} \times \omega_{C_i,F_j}\right) \quad (8)$$

Similarly, communication cost from fog to cloud offloading denoted as $\tau^{F\to C}$ will be

$$\tau^{F\to C} = \sum_{i=1}^{m} \sum_{j=1}^{n} \left(O_{F_j,C_i} \times \omega_{C_i,F_j}\right) \quad (9)$$

Finally, the fog to fog communication cost denoted as $\tau^{F\to F}$ can be expressed as

$$\tau^{F\to F} = \sum_{j=1}^{n} \left(\left(\sum_{j'=1}^{n} O_{F_j,F_{j'}} + \sum_{j'=1}^{n} O_{F_{j'},F_j}\right) \times \omega_{F_i,F_{j'}}\right) \quad (10)$$

Along with the communication and computing cost, we introduce another cost that is called the waiting cost. The waiting cost appears in a special scenario, where requests arrive at a node, but neither get the required resources for processing nor get offloaded to other nodes. The total waiting cost for cloud, and fog denoted as τ_{w_C} and τ_{w_F}, respectively, are expressed as

$$\tau_{w_C} = \sum_{i=1}^{m} \left(W_{C_i} \times \mu_{C_i}^E \times S_{C_i}\right) \quad (11)$$

$$\tau_{w_F} = \sum_{j=1}^{n} \left(W_{F_j} \times \mu_{F_j}^E \times S_{F_j}\right) \quad (12)$$

The total waiting cost τ_{wait}, total communication cost τ_{comm}, and total computing cost τ_{comp} can be as

$$\tau_{wait} = \tau_{w_C} + \tau_{w_F}, \quad (13)$$

$$\tau_{comm} = \tau^{C\to F} + \tau^{F\to C} + \tau^{F\to F}, \quad (14)$$

$$\tau_{comp} = \tau_C + \tau_F, \quad (15)$$

Finally, the main objective is to minimize the total cost in the system denoted as $\tau_{total} = \tau_{comm} + \tau_{comp} + \tau_{wait}$ expressed as

$$\min \tau_{total}, \quad (16)$$

subject to,

$$L_{C_i} < L_{MAX}, \quad (17)$$

$$L_{F_j} < L_{MAX}, \quad (18)$$

$$L_{C_i,F_j} + L_{F_j} < L_{MAX} \ , \quad (19)$$
$$L_{F_j,F_{j'}} + L_{F_{j'}} < L_{MAX} \ , \quad (20)$$
$$L_{F_j,C_i} + L_{C_i} < L_{MAX} \ , \quad (21)$$
$$\mu_{C_i}^{U} < \mu_{C_i}^{MAX} \ , \quad (22)$$
$$\mu_{F_j}^{U} < \mu_{F_j}^{MAX} \ , \quad (23)$$
$$O_{C_i,F_j} < B_{C_i,F_j} \ , \quad (24)$$
$$O_{F_j,C_i} < B_{C_i,F_j} \ , \quad (25)$$
$$O_{F_j,F_{j'}} < B_{F_j,F_{j'}} \ , \quad (26)$$

Equations (17) and (18) ensure that the request coming to the cloud and fog being processed at the same cloud and fog, respectively, do not exceed the maximum latency L_{MAX}. The constraint (19) ensures that the request arriving at C_i and later on being executed at F_j do not exceed the maximum latency limit. Equations (20), and (21) ensure that the request arriving at F_j and executed at $F_{j'}$, and C_i, respectively, do not exceed the maximum latency. The capacity constraints in equations (22), and (23) ensure that the capacity in use in the cloud, and fog, respectively, do not exceed the respective maximum computing capacity. Equations (24), (25), and (26) ensure that the offloading request from cloud to fog, fog to the cloud, and fog to fog, respectively should not exceed the respective channel bandwidth.

4. ALGORITHM DESIGN

In this paper, we used a modified simulated annealing (SA) algorithm to handle the two-tier cloud-fog federated systems. The SA algorithm flow starts with setting up the initial and final temperatures which are set to 100 and 10, respectively. The cooling parameter α is set to 0.99. From the default parameter, we generate the solution $\mathbb{S}$ and check whether they satisfy the given latency and capacity constraints. In the next step, we generate a new solution $\mathbb{S}_{new}$ that is explained in *Algorithm 2*. Then from $\mathbb{S}$ and $\mathbb{S}_{new}$ we compute the total computation cost of cloud and fog. Then estimate Δ_C, and Δ_F as given in Line 6. From Line 7-15, we compare the old solution with the new solution. If the new solution is better, we accept it, if not, we still accept it with the probability given in Line 10, 12, and 14. In each iteration, we reduce the initial temperature by multiplying the α. The process will continue until it satisfies the condition given in Line 17.

In *Algorithm 2*, we demonstrate the procedure of generating the new solution. It is executed in a loop that leads to checking three conditions. The first condition checks that the required capacity to be in use exceeds the maximum capacity of the fog. The neighboring fog has available capacity to process the requests; the task will be offloaded to the neighboring fog by *Algorithm 3*, provided it will not violate the given latency constraint. If the first condition does not satisfy, we will check the second condition i.e., the neighboring fog would not be able to process the requests and will be offloaded to the cloud by *Algorithm 4*. In the last condition, when the fog has enough capacity to process the request, the cloud will offload its requests by *Algorithm 5* as the processing in the fog is cheaper, which helps reduce the cost.

The fog to fog offloading scenario is explained in *Algorithm 3*. It ensures that requests from one fog would not be offloaded to the same fog. How many requests are offloaded is determined by value γ, which is the offloading ratio with a range from 0 to 0.1. The offloading ratio would be multiplied by the capacity in that fog to obtain the total request to be offloaded to other fog.

Algorithm 1 Modified Simulated Annealing

1: Set T = 100, T_{end} = 10 and α = 0.99
2: Generate $\mathbb{S}$ from default parameter
3: Check $\mathcal{L}$ and $\mathbb{C}$ for initial solution
4: Generate $\mathbb{S}_{new}$ by Algorithm 2
5: Compute the old τ_C and τ_F from $\mathbb{S}$, and new τ_C and τ_F from $\mathbb{S}_{new}$
6: Calculate Δ_C by subtracting new τ_C and old τ_C, and Δ_F by subtracting new τ_F and old τ_F
7: **if** $\Delta_C \leq 0$ and $\Delta_F \leq 0$ **then**
8: Accept $\mathbb{S}_{new}$
9: **else if** $\Delta_C \leq 0$ and $\Delta_F > 0$ **then**
10: Accept $\mathbb{S}_{new}$ with probability $\exp\left(-\frac{\Delta_F}{T}\right)$
11: **else if** $\Delta_C > 0$ and $\Delta_F \leq 0$ **then**
12: Accept $\mathbb{S}_{new}$ with probability $\exp\left(-\frac{\Delta_C}{T}\right)$
13: **else**
14: Accept $\mathbb{S}_{new}$ with probability $\exp\left(-\frac{\Delta_C+\Delta_F}{T}\right)$
15: **end if**
16: Compute $T \leftarrow T * \alpha$
17: **if** $T < T_{end}$ **then**
18: Return $\mathbb{S}_{new}$
19: **else**
20: go back to step 4
21: **end if**

Algorithm 2 Generate new solution

Require: $\mu_{C_i}^{U}, \mu_{F_j}^{U}, \mu_{F_j}^{MAX} \mathcal{L}, \mathbb{C}, F_j, C_i$

1: **while** $\mathbb{S}_{new}$ fail $\mathcal{L}$ and $\mathbb{C}$ **do**
2: **if** $\mu_{F_j}^{U} > \mu_{F_j}^{MAX}$ and $\mu_{F_j}^{U} < \mu_{F_{j'}}^{MAX}$ and high $\mathcal{L}$ **then**
3: Offload fog to fog by Algorithm 3
4: **else if** $\mu_{F_j}^{U} > \mu_{F_j}^{MAX}$ and $\mu_{F_j}^{U} > \mu_{F_{j'}}^{MAX}$ or low $\mathcal{L}$ **then**
5: Offload fog to cloud by Algorithm 4

6: **else if** $\mu_{F_j}^U < \mu_{F_j}^{MAX}$ **then**
7: Offload cloud to fog by Algorithm 5
8: **end if**
9: **end while**
10: **return** $\mathbb{S}_{new}$

Algorithm 3 Offloading Fog-to-Fog

Require: $\mu_{F_j}^U, \mu_{F_j}^{MAX}, F_j$
1: **for** $j \leftarrow 0$ to F_j **do**
2: **for** $j' \leftarrow 0$ to F_j **do**
3: Generate $\gamma \leftarrow 0$
4: **if** $j \neq j'$ and $\mu_{F_j}^U > \mu_{F_j}^{MAX}$ **then**
5: $\gamma \leftarrow random(0, 0.1)$
6: $O_{F_{j,F_{j'}}} \leftarrow O_{F_{j,F_{j'}}} + \mu_{F_j}^U * \gamma$
7: $\mu_{F_j}^U \leftarrow \mu_{F_j}^U - O_{F_{j,F_{j'}}}$
8: $\mu_{F_{j'}}^U \leftarrow \mu_{F_{j'}}^U + O_{F_{j,F_{j'}}}$
9: **end if**
10: **end for**
11: **end for**

Algorithm 4 Offloading Fog-to-Cloud

Require: $\mu_{C_i}^U, \mu_{F_j}^U, \mu_{F_j}^{MAX}, F_j, C_i$
1: **for** $i \leftarrow 0$ to C_i **do**
2: **for** $j \leftarrow 0$ to F_j **do**
3: Generate $\gamma \leftarrow 0$
4: **if** $\mu_{F_j}^U > \mu_{F_j}^{MAX} * 0.5$ **then**
5: $\gamma \leftarrow random(0.01, 0.1)$
6: **else**
7: $\gamma \leftarrow random(0, 0.01)$
8: **end if**
9: $O_{F_{j,C_i}} \leftarrow O_{F_{j,C_i}} + \mu_{F_j}^U * \gamma$
10: $\mu_{C_i}^U \leftarrow \mu_{C_i}^U + O_{F_{j,C_i}}$
11: $\mu_{F_j}^U \leftarrow \mu_{F_j}^U - O_{F_{j,C_i}}$
12: **end for**
13: **end for**

Algorithm 5 Offloading Cloud-to- Fog

Require: $\mu_{C_i}^U, \mu_{F_j}^U, \mu_{F_j}^{MAX}, F_j, C_i$
1: **for** $i \leftarrow 0$ to C_i **do**
2: **for** $j \leftarrow 0$ to F_j **do**
3: Generate $\gamma \leftarrow 0$
4: **if** $\mu_{F_j}^U > \mu_{F_j}^{MAX} * 0.5$ **then**
5: $\gamma \leftarrow random(0, 0.01)$
6: **else**
7: $\gamma \leftarrow random(0.01, 0.1)$
8: **end if**
9: $O_{C_i,F_j} \leftarrow O_{C_i,F_j} + \mu_{C_i}^U * \gamma$
10: $\mu_{C_i}^U \leftarrow \mu_{C_i}^U - O_{C_i,F_j}$
11: $\mu_{F_j}^U \leftarrow \mu_{F_j}^U + O_{C_i,F_j}$
12: **end for**
13: **end for**

The fog to cloud offloading scenario is explained in *Algorithm 4*. When the fog's in-use computing capacity is fifty percent above the maximum computing capacity, it will offload more requests to the cloud, and when it is less than fifty percent, it offloads less. The value γ determines the offloading ratio in both cases.

The cloud to fog offloading scenario is explained in *Algorithm 5*. When the fog's in-use computing capacity is fifty percent above the maximum computing capacity, the cloud will offload fewer requests to the fog, and when it is less than fifty percent, it offloads more. Here, like *Algorithm 4*, the value γ determines the offloading ratio in both cases.

5. PERFORMANCE ANALYSIS

In this section, we will discuss the performance of our proposed architecture and algorithm. The parameters used for the simulation are presented in Table II.

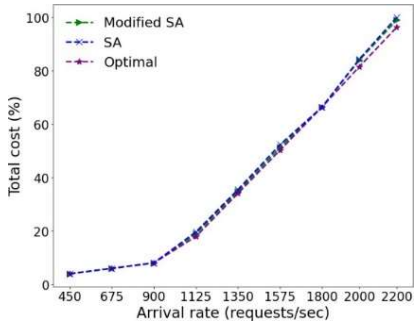

Figure. 5: Impact of different algorithms on total cost.

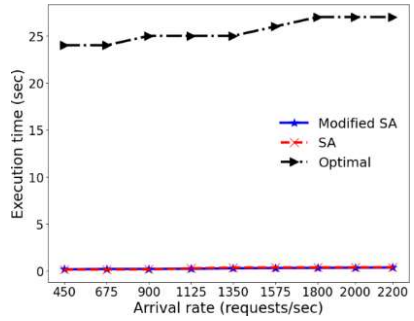

Figure 6: Impact of different algorithms on execution time compared to optimal solution.

Table II: Simulation Parameters

Notation	Meaning	Value	Unit
n	Number of clouds	4	n/a
m	Number of fogs	10	n/a
μ_{Ci}^{MAX}	Computation Capacity of cloud	1000	CPU Cycles
μ_{Fj}^{MAX}	Computation Capacity of fog	100	CPU Cycles
$B_{Ci,Fj}$	Communication capacity between cloud and fog	500	Bytes/request
$B_{Fj,Fj'}$	Communication capacity between fog and fog	100	Bytes/request
φ_{Ci}	Cloud computing cost	100	Money units/cycles
φ_{Fj}	Fog computing cost	10	Money units/cycles
$\omega_{Ci,Fj}$	Communication cost between cloud and fog	5	Money units/bytes
$\omega_{Fj,Fj'}$	Communication cost between fog and fog	1	Money units/bytes
S_{Ci}	Cloud waiting cost	200	Money units/ms
S_{Fj}	Fog waiting cost	20	Money units/ms
L_{MAX}	Maximum latency	0.1, 0.5, 1, 10, 100, 1000	ms

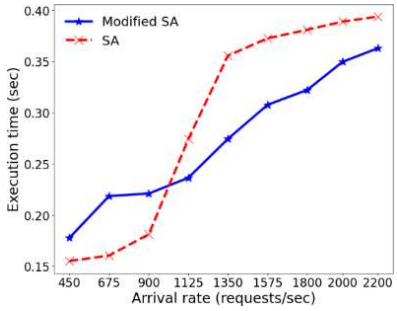

Figure 7: Impact of simulated annealing and modified simulated annealing algorithm.

Fig. 5 shows with the increase in arrival traffic, the total cost increases where the maximum latency limit was set to 1000ms. The performance of the proposed modified SA and SA are nearly similar to optimal results that are obtained using the IBM CPLEX [36]. Although the total cost of the algorithms is similar to the optimal results, their execution time is significantly better than the CIPLEX optimizer. As Fig. 6 shows, while the optimizer takes more than 25 seconds to produce the optimal results, the simulation can be done in less than 1 sec. Although the execution time of SA and modified SA look the same in Fig.6, Fig.7 demonstrates their performance difference. The result in Fig. 7 shows with the increase in traffic, the modified SA outperforms the SA.

Fig. 8 shows the performance of the proposed OMNI architecture in different latency constraints. As the result shows, with the tight latency, the cost is high, and the cost goes down with the loose latency. Because we know the cloud has low computing latency because of its high-power computing capacity but with a very high cost. Whereas the fog has high computing latency due to its low power computing resources with a lower cost. However, it adds extra communication latency when tasks are offloaded from fog to cloud or cloud to fog. Hence in loose latency, execution is done where computation is cheap, whereas, for tight latency, tasks are offloaded to one that can serve faster.

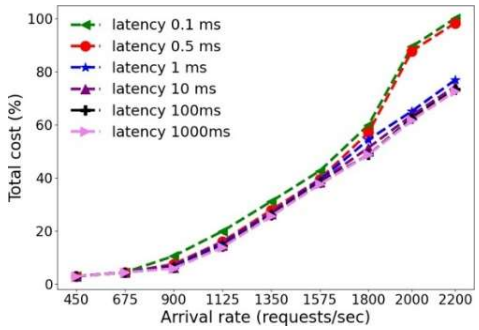

Figure 8: Impact of different latency on the total cost.

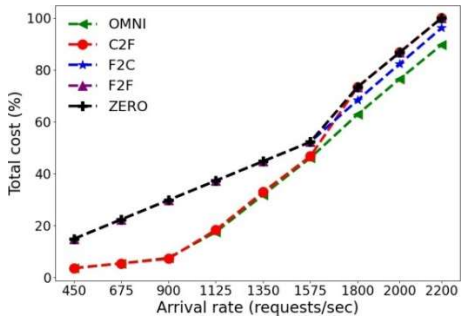

Figure 9: Impact of different architectures on total cost with latency 100ms.

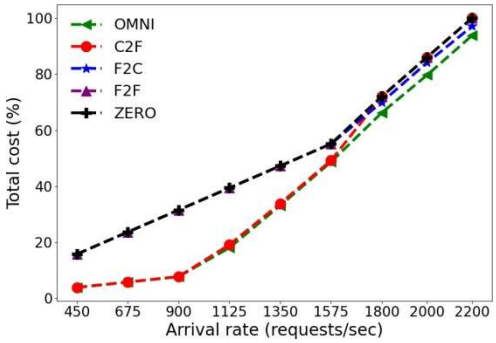

Figure 10: Impact of different architectures on total cost with latency 1000ms.

In Fig. 9, and 10, we compare the performance of our OMNI architecture with cloud to fog (C2F), fog to cloud (F2C), fog to fog(F2F), and no offloading (ZERO) models. In Fig. 9, the latency limit is set to 100ms, and in Fig. 10 is 1000ms. As both the results show, our OMNI architecture has a lower cost than all other architectures. This is because it has the option to offload the tasks in all possible directions to where it can lower the cost without violating the given latency constraint.

6. CONCLUSION

In this work, we proposed a cloud-fog federated OMNI architecture and investigated the cost minimization problem with given latency limits. We used a modified SA algorithm as a solution. The results show the proposed algorithm performance is nearly optimal. It outperforms the SA algorithm in terms of execution time. We also compared the performance of the proposed OMNI architecture with other architectures. The results show that the OMNI architecture saves more cost than other architectures in different latency limits.

REFERENCES

[1] B. Furht, A. Escalante et al., *Handbook of cloud computing*. Springer,2010, vol. 3.

[2] M. Othman, S. A. Madani, S. U. Khan et al., "A survey of mobile cloud computing application models," *IEEE communications surveys & tutorials*, vol. 16, no. 1, pp. 393–413, 2013.

[3] M. Mukherjee, L. Shu, and D. Wang, "Survey of fog computing: Fundamental, network applications, and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 1826–1857, 2018.

[4] R. Mahmud, R. Kotagiri, and R. Buyya, "Fog computing: A taxonomy, survey and future directions," in *Internet of everything*. Springer, 2018, pp. 103–130.

[5] G. Zhang, F. Shen, Z. Liu, Y. Yang, K. Wang, and M.-T. Zhou, "Femto: Fair and energy-minimized task offloading for fogenabled iot networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4388–4400, 2018.

[6] S. El Kafhali and K. Salah, "Efficient and dynamic scaling of fog nodes for iot devices," *The Journal of Supercomputing*, vol. 73, no. 12, pp. 5261–5284, 2017.

[7] L. Liu, Z. Chang, X. Guo, S. Mao, and T. Ristaniemi, "Multiobjective optimization for computation offloading in fog computing," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 283–294, 2017.

[8] C. A. Lee, R. B. Bohn, M. Michel, A. Delaitre, B. Stivalet, and P. E. Black, "The nist cloud federation reference architecture 5," *NIST Special Publication*, vol. 500, p. 332, 2020.

[9] G.-E. Zaharia, R.-I. Ciobanu, C. Dobre et al., "Machine learning based traffic offloading in fog networks," *Simulation Modelling Practice and Theory*, vol. 101, p. 102045, 2020.

[10] Q. Wu, H. Ge, H. Liu, Q. Fan, Z. Li, and Z. Wang, "A task offloading scheme in vehicular fog and cloud computing system," *IEEE Access*, vol. 8, pp. 1173–1184, 2019.

[11] Z. Wei and H. Jiang, "Optimal offloading in fog computing systems with non-orthogonal multiple access," *IEEE Access*, vol. 6, pp. 49 767–49 778, 2018.

[12] J. Wang, J. Pan, F. Esposito, P. Calyam, Z. Yang, and P. Mohapatra, "Edge cloud offloading algorithms: Issues, methods, and perspectives," *ACM Computing Surveys (CSUR)*, vol. 52, no. 1, pp. 1–23, 2019.

[13] H. Shah-Mansouri and V. W. Wong, "Hierarchical fog-cloud computing for iot systems: A computation offloading game," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 3246–3257, 2018.

[14] X. Meng, W. Wang, and Z. Zhang, "Delay-constrained hybrid computation offloading with cloud and fog computing," *IEEE Access*, vol. 5, pp. 21 355–21 367, 2017.

[15] L. Liu, Z. Chang, and X. Guo, "Socially aware dynamic computation offloading scheme for fog computing system with energy harvesting devices," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1869–1879, 2018.

[16] H. Lin, S. Zeadally, Z. Chen, H. Labiod, and L. Wang, "A survey on computation offloading modeling for edge computing," *Journal of Network and Computer Applications*, vol. 169, p. 102781, 2020.

[17] Y.-L. Jiang, Y.-S. Chen, S.-W. Yang, and C.-H.Wu, "Energy-efficient task offloading for time-sensitive applications in fog computing," *IEEE Systems Journal*, vol. 13, no. 3, pp. 2930–2941, 2018.

[18] Y. Jiang and D. H. Tsang, "Delay-aware task offloading in shared fog networks," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4945–4956, 2018.

[19] J. Du, L. Zhao, J. Feng, and X. Chu, "Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee," *IEEE Transactions on Communications*, vol. 66, no. 4, pp. 1594–1608, 2017.

[20] B. Wang, C. Wang, W. Huang, Y. Song, and X. Qin, "A survey and taxonomy on task offloading for edge-cloud computing," *IEEE Access*, vol. 8, pp. 186 080–186 101, 2020.

[21] M. Arif, F. Ajesh, S. Shamsudheen, and M. Shahzad, "Secure and energy-efficient computational offloading using lstm in mobile edge computing," *Security and Communication Networks*, vol. 2022, 2022.

[22] B. Kar, W. Yahya, Y.-D. Lin, and A. Ali, "A survey on offloading in federated cloud-edge-fog systems with traditional optimization and machine learning," *arXiv preprint arXiv:2202.10628*, 2022.

[23] L. Mashayekhy, M. M. Nejad, and D. Grosu, "Cloud federations in the sky: Formation game and mechanism," *IEEE Transactions on Cloud Computing*, vol. 3, no. 1, pp. 14–27, 2014.

[24] H. Chen, B. An, D. Niyato, Y. C. Soh, and C. Miao, "Workload factoring and resource sharing via joint vertical and horizontal cloud federation networks," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 3, pp. 557–570, 2017.

[25] X. Cao, G. Tang, D. Guo, Y. Li, and W. Zhang, "Edge federation: Towards an integrated service provisioning model," *IEEE/ACM Transactions on Networking*, vol. 28, no. 3, pp. 1116–1129, 2020.

[26] M.-T. Thai, Y.-D. Lin, Y.-C. Lai, and H.-T. Chien, "Workload and capacity optimization for cloud-edge computing systems with vertical and horizontal offloading," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 227–238, 2019.

[27] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7944–7956, 2019.

[28] Y. Wen, Q. Zhang, H. Yuan, and J. Bi, "Multi-stage pso-based cost minimization for computation offloading in vehicular edge networks," in 2021 *IEEE International Conference on Networking, Sensing and Control (ICNSC),* vol. 1. IEEE, 2021, pp. 1–6.

[29] Q. Luo, W. Shi, and P. Fan, "Qoe-driven computation offloading: Performance analysis and adaptive method," *in 2021 13th International Conference on Wireless Communications and Signal Processing (WCSP).* IEEE, 2021, pp. 1–5.

[30] J. Bi, H. Yuan, K. Zhang, and M. Zhou, "Energy-minimized partial computation offloading for delay-sensitive applications in heterogeneous edge networks," *IEEE Transactions on Emerging Topics in Computing*, 2022.

[31] M. Aazam, S. Zeadally, and K. A. Harras, "Offloading in fog computing for iot: Review, enabling technologies, and research opportunities," *Future Generation Computer Systems*, vol. 87, pp. 278–289, 2018.

[32] H. Mahini, A. M. Rahmani, and S. M. Mousavirad, "An evolutionary game approach to iot task offloading in fog-cloud computing," *The Journal of Supercomputing*, vol. 77, no. 6, pp. 5398–5425, 2021.

[33] R.-H. Hwang, Y.-C. Lai, and Y.-D. Lin, "Offloading optimization with delay constraint in the 3-tier federated cloud, edge, and fog systems," in 2021 *IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2021, pp. 1–6.

[34] R. Besharati and M. H. Rezvani, "A prototype auction-based mechanism for computation offloading in fog-cloud environments," *in 2019 5th conference on knowledge based engineering and innovation (KBEI)*. IEEE, 2019, pp. 542–547.

[35] B. Kar, Y.-D. Lin, and Y.-C. Lai, "Omni: Omni-directional dual cost optimization of two-tier federated cloud-edge systems," *in ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–7.

[36] C. U. Manual, "Ibm ilog cplex optimization studio," *Version*, vol. 12, pp. 1987–2018, 1987. [Online]. Available: http://www.ibm.com/software/integration/optimization/cplex-optimizer

[37] Akutsu, K., Phung-Duc, T., Lai, Y.C. and Lin, Y.D., 2022. Analyzing vertical and horizontal offloading in federated cloud and edge computing systems. Telecommunication Systems, 79(3), pp.447-459.