HeRAFC: Heuristic Resource Allocation and Optimization in MultiFog-Cloud Environment

Chinmaya Kumar Dehury[a], Bharadwaj Veeravalli[b], Satish Narayana Srirama[c,*]

[a]*Mobile & Cloud Lab, Institute of Computer Science, University of Tartu, Tartu 50090, Estonia*
[b]*Department of Electrical and Computer Engineering, National University of Singapore, Singapore.*
[c]*School of Computer and Information Sciences, University of Hyderabad, Gachibowli, Telangana, India.*

Abstract

By bringing computing capacity from a remote cloud environment closer to the user, fog computing is introduced. As a result, users can access the services from more nearby computing environments, resulting in better quality of service and lower latency on the network. From the service providers' point of view, this addresses the network latency and congestion issues. This is achieved by deploying the services in cloud and fog computing environments. The responsibility of service providers is to manage the heterogeneous resources available in both computing environments. In recent years, resource management strategies have made it possible to efficiently allocate resources from nearby fog and clouds to users' applications. Unfortunately, these existing resource management strategies fail to give the desired result when the service providers have the opportunity to allocate the resources to the users' application from fog nodes that are at a multi-hop distance from the nearby fog node. The complexity of this resource management problem drastically increases in a MultiFog-Cloud environment. This problem motivates us to revisit and present a novel Heuristic Resource Allocation and Optimization algorithm in a MultiFog-Cloud (HeRAFC) environment. Taking users' application priority, execution time, and communication latency into account, HeRAFC optimizes resource utilization and minimizes cloud load. The proposed algorithm is evaluated and compared with related algorithms. The simulation results show the efficiency of the proposed HeRAFC over other algorithms.

Keywords: Fog computing, cloud computing, resource management, resource allocation, MultiFog-Cloud, heuristic algorithm

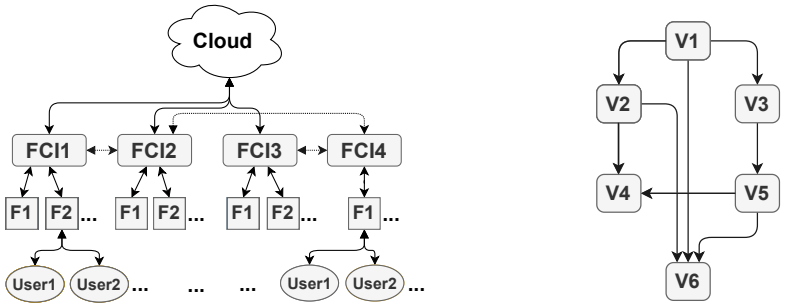
1. Introduction

In a bottom-up approach, the Fog Node (FN) receives the users' requests consisting of multiple tasks. FN may process some tasks of a request and offload others to the cloud environment based on the resource availability at the FN, the Service Level Agreement (SLA), the priority of the user or request, and other Quality of Service (QoS) requirements. However, in a top-down approach, the resource availability in the cloud environment is first checked to execute the tasks. Following this, the rest of the tasks of a user's request are assigned to the available resources at FN [1, 2]. Both approaches differ based on what environment is given higher preference while allocating the resource to the users' request. Resource in this context refers to CPU, memory, storage, processing capability, etc., of the servers and bandwidth and transmission latency of the network.

The current architectures consist of multiple users at the bottom layer, served by single FN at the fog layer (middle layer) and a single cloud at the top of the hierarchy [3, 4, 5]. The FNs are consisting of less amount of computing and storage resource. On the other hand, from the user's perspective, the resource availability at the cloud is generally considered as infinite, thanks to the underlined virtualization technology and other resource management tools. Management of the entire lifecycle of such diverse computing environments with different sets of resource capacity, configurations and characteristics is refereed to as heterogeneous resource management [6]. The heterogeneous resource management for single FN and single cloud environments is crucial especially while hosting the user's request. Inefficient management may lead to resource unavailability at FN, prohibiting the task from being deployed on FN. Several approaches, such as Integer Linear Programming (ILP) [4], heuristic approach, genetic algorithm, game theory [5], etc., have been investigated recently to optimize resource utilization. This resource management problem becomes more complex

*Corresponding author.
Email addresses: chinmaya.dehury@ut.ee (Chinmaya Kumar Dehury), elebv@nus.edu.sg (Bharadwaj Veeravalli), satish.srirama@uohyd.ac.in (Satish Narayana Srirama)

when the multiple interconnected FNs are introduced at the middle layer of the hierarchy, as shown in Figure 1a [7]. This paper addresses the resource allocation and optimization problem in the above-mentioned multiFog-Cloud environment.



(a) Abstract view of MultiFog-Cloud (MFC) environment.

(b) An example of user's application (in DAG form).

Figure 1: Example of MFC and user's application.

1.1. Motivation and Contribution

The existing mechanisms classify the application tasks into two groups: tasks for fog environment and tasks for the cloud environment, based on the resource demand, task type, priority, deadline of the application task, etc. As per the classification result, tasks are deployed and the resources are allocated to the tasks either from nearby fog or the cloud system. For example, in the case of real-time video streaming analysis in smart city, the raw video is preprocessed in the nearby fog computing node. Further, the preprocessed video is forwarded to cloud for in-detail processing, indexing and time-stamping of the video, which can be stored and accessed in future across the globe. In this fog-cloud architecture, single FN and one or more cloud providers are involved.

The above fog-cloud architecture works efficiently as the multiple FNs across the street are deployed to handle the growing demand. However, due to uneven workload demand among the FNs in the city, some tasks may get rejected by the FNs at the city center and the request may get forwarded to the cloud. In such case, it is hard to maintain the QoS and the service provider may violate the Service Level Agreement (SLA). On the other hand, the resources of the FNs that receive less number of demand, remain underutilized, resulting

in lesser revenue for the service provider. Here, revenue refers to the monetary income generated from the service provider's business. In order to maintain the higher revenue, the service provider may increase the service cost affecting the users' QoE. This motivates us to revisit the resource allocation problem in a fog-cloud environment with the goal to maximize the resource utilization of the FNs.

In order to address above research issues and be able to provide services in a real-time manner by the fog-cloud computing environment, in this paper, we attempt to modify the existing fog-cloud computing model. This is done via establishing the communication among the FNs through Fog-Cloud Interface (FCI), as shown in Figure 1a. The scope of this paper is to meet the following main goals which are summarized as follows:

- Design a novel resource allocation strategy for MFC environment and multi-task user application.
- To optimize the resource utilization by allowing the tasks to be placed onto nearby FNs that are at multi-hop distance.
- To design and evaluate the performance of an algorithm to decide the order of processing and assigning the tasks onto MFC environment by considering the tasks' topology and their resource demand.

Based on the motivation and the goals mentioned above, the main contributions in this paper can be summarized as follows:

- An architecture for the multifog-cloud is designed that allows application tasks to be forwarded to other FNs at multi hop distance, thus reducing the workload on the cloud.
- The problem of resource allocation in multifog-cloud scenario is formulated as Integer Linear Programming (ILP) model. FNs located at multi-hop distances can also handle the users' requests, which complicates the problem when the different FCIs have different communication capabilities.

Table 1: List of Acronyms

Acronym	Description	Acronym	Description
CPS	Cyber Physical System	QoS	Quality of Service
DAG	Directed Acyclic Graph	SFC	Service Function Chain
FCI	Fog-Cloud Interface	SLA	Service Level Agreement
FN	Fog Node	VM	Virtual Machine
ILP	Integer Linear Programming	WMD	Weighted Multi-Dimensional
IoT	Internet of Things	LIFO	Last-In-First-Out
MFC	MultiFog-Cloud	FCFS	First Come, Fist Served
QoE	Quality of Experience		

- The physical infrastructure of multifog-cloud is modeled using graph theory.
- A novel dedicated heuristic algorithm is designed for placement of the application tasks in multifog-cloud scenario.
- We exhibit the performance of proposed algorithms in improving fog and cloud computing resource utilization, network utilization, latency improvement, and other system parameters through extensive simulation.

The rest of the paper is organized as follows. In Section 2, a brief summary of recent related articles is presented. In Section 3, the system model on the users' application and physical infrastructure is presented. Section 4 presents the formulation of resource allocation problem in single-Fog single-cloud environment, where the user's tasks are deployed either in nearby FN or in cloud node. This followed by Section 5, formulation of resource allocation problem in Multi-Fog Cloud environment, where user's tasks will have more options to be deployed. Considering the model, a heuristic resource allocation strategy is presented in Section 6 followed by the performance evaluation of the proposed algorithm in Section 7. The concluding remarks and the scope of the future work are presented in Section 8. The list of acronyms that are used in this article is given in Table 1.

2. Related Works

In this section, the recent related research articles are reviewed and presented taking different aspects into account. The resource management problem of fog and cloud computing is addressed in general, such as in [8, 1, 9, 10, 11, 2].

2.1. Resource management in fog-cloud

Authors in [4, 5] address the resource allocation problem with the objective to reduce the computation energy and the delay. The proposed computation offloading scheme in [4] allows multiple end-users to transmit the data to the same nearby FN. However, the proposed algorithm may not be suitable when the cloud and fog environments are combined. Similarly, in [5] the game theory approach is followed to allocate the fog-cloud resources among IoT users with the goal to maximize the number of users, energy cost and the overall delay. The same resource allocation problem can also be seen and approached by using priced timed Petri nets method, as in [12], which predicts the task completion time and proactively allocate the resource in a dynamic manner considering both the monetary cost and task completion cost. The major pitfall of above resource allocation mechanisms is their inability to handle multi fog scenario. At a higher level of resource management problem, authors in [13] proposed a resource identity management strategy considering fog and cloud computing environment. This would help the system administrator to manage not only the cloud resource but also the fog resources that are provided by the end-users. The limitation with this approach lies within single fog to cloud communication, which may not fit to the today's scenarios.

The proposed placement strategy in [14] uses First-Fit approach to find the suitable FN for each parallel service modules. Extending the proposed scheme to fit the multi-fog scenario could further increase the concurrency of the service execution. Similarly, in [15], authors proposed the application placement strategy in fog computing environment taking the Quality of Experience (QoE) into account. The proposed mechanism prioritizes the users' applications based on their requirements, intention, execution platform and other users' expectation parameters, which may slightly deviate from the QoS parameters. Authors in [10] investigated the resource allocation problem from the security point of view and proposed privacy-preserving resource allocation mechanisms for fog environments. The proposed mechanism mainly revolves around the fog computing environment and may not give efficient result in case of cloud environment.

Introducing the fog computing environment between the industrial cloud and the ter-

minal devices, authors in [16] proposed a task scheduling algorithm, that ensures the task completion time and optimizes the execution concurrency. However, the proposed algorithm lacks the ability to utilize the computing resource available at the terminal of the nearby fog devices. Similarly, considering the heterogeneous FNs, a scalable and decentralized scheduling algorithm is presented by authors in [17] with the goal to minimize the service delay by efficiently managing computation and communication resources of FNs.

2.2. Energy optimization

In order to offload the computation, while minimizing the energy consumption, authors in [18, 19] proposed different strategies, that mitigate and handle the problem of growing resource demand of IoT users. In [18], authors optimize the three major parameters: transmission power of the diverse application, computation resource distribution among those application and offloading decision. Similarly, authors in [19] optimize two QoS parameters: energy consumption and computational time in sustainable fog-cloud infrastructure. To meet the requirement of multi-fog and cloud environment the above proposed mechanisms need to be extended significantly.

Based on the cost efficiency, authors in [20] investigated the cloud and fog resource management problem. The problem is formulated and presented in a three-layer computing environment. A double two-sided matching optimization model is designed keeping the high cost-efficiency performance in mind. Similarly, with the min-max fairness guarantee, authors in [21] proposed a sub-optimal resource allocation strategy that offloads the computational task to minimize the energy consumption and delay cost. The strategy is to divide the entire problem into offloading decision-making problem and resource allocation problem, which are solved by semidefinite relaxation and randomization method and Lagrangian dual decomposition method, respectively.

2.3. Real-world application specific

The resource management problem is also addressed considering diverse practical real-life applications such as vehicular network [22, 4], smart grid [23, 24], smart Buildings

[25, 26], smart manufacturing [16, 27], smart city [28, 29]. Authors in [22] presented an adaptive resource management algorithm for vehicular networks with the goal to minimize the transmission rate, delay-jitter and the upper-bound of delay. A model for the integration of fog and cloud with smart grid is presented in [23], where the data flow and the request forwarding for electricity to micro-grid are handled by FNs. Fog and cloud computing environments are used for management of the smart building resources through different load balancing algorithms in [25, 26, 30, 13]. From the application point of view, authors in [28] show how the smart city resources can be managed by taking advantage of fog computing. The fog computing model acts as the buffer and controller between cyber-physical world and the cloud computing environment with the goal to reduce the coupling in computing and maximize the utilization of resources.

3. System Model

The users' request consists of multiple inter-connected tasks. An example of user's request consisting of six tasks is shown in Figure 1b. Each task is further associated with a certain amount of resource demand, fulfilled by the servers at fog and cloud environments. Figure 1a shows the proposed hierarchy of users base at the bottom, FNs and cloud environment. Users access the services from the nearby FNs at the fog layer. For example, *User1* and *User2* access the requested service from the FN *F1*. Between the fog layer and the cloud environment, a Fog-Cloud Interface (FCI) is introduced. FCI is responsible for (1) establishing a communication bridge between FNs and cloud and (2) establishing communication interface among FNs. It is also assumed that the FCIs may be interconnected among themselves. The communication among the FCIs enables the communication among FNs. As presented in Figure 1a, FNs that are connected to *FCI1* may communicate with the FNs that are connected to *FCI4* through *FCI2*. This allows the users, connected to FN *F1* under *FCI4*, to access the cloud resource through *FCI2*.

The placement and resource allocation strategy follows a bottom-up approach. As a result, the users' tasks are first forwarded to the fog environment. If the fog resources are not enough to fulfill the demand, the users' tasks are further forwarded to cloud. Due to

Table 2: List of key notations and description

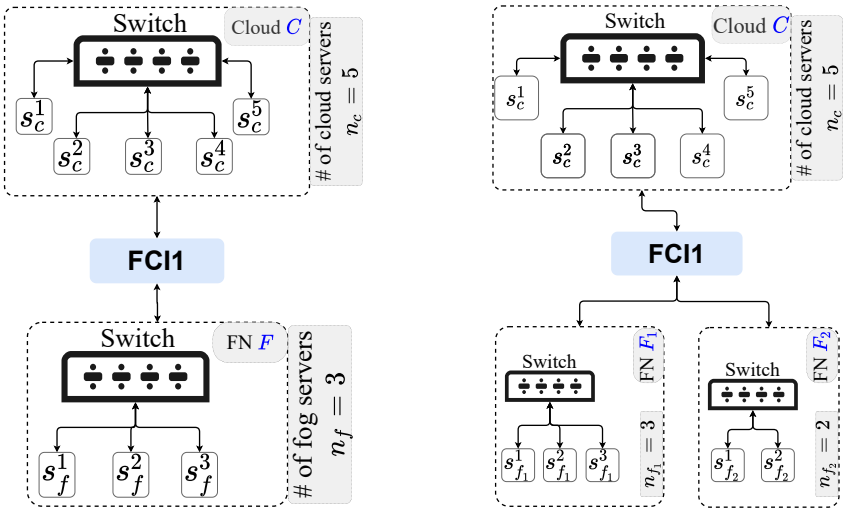
Notation	Description
G	$= (V, E)$ Resource graph
V	$= \{C, F_1, F_2, \dots\}$ in MFC environment
C	Set of servers in cloud environment $= \{s_c^1, s_c^2, s_c^3, .., s_c^{n_c}\}$
F_i	FN i in MFC environment
$s_{f_j}^i$	Server i in FN F_j
$\vec{P}_{ij}$	Physical path between server $s^{m_i} \in F_k$ and $s^{m_j} \in F_l, k \neq l$
$\vec{H}_{ij}$	Number of hops present in between the server $s^{m_i} \in F_k$ and $s^{m_j} \in F_l, k \neq l$
$R^x(s^{m_1})$	Remaining resource of type x available at server $s^{m_1} \in V$
$\alpha(s^{m_1}, s^{m_2})$	Available bandwidth available between server $s^{m_1} \in V$ and $s^{m_2} \in V$
$\beta(s^{m_1}, s^{m_2})$	Network latency on the physical link between server $s^{m_1} \in V$ and $s^{m_2} \in V$
G'	$= (V', E')$ User's application
V'	Set of tasks in user's application. $V' = \{v_1, v_2, ..., v_{n'}\}$
E'	Set of edges between users' tasks
$\hat{e}(v_i, v_j)$	The boolean variable indicating if there is an edge existing from task v_i to task v_j
$\bar{R}^x(v_i)$	Resource demand of type x by task $v_i \in V'$
$\alpha'(v_i, v_j)$	Bandwidth demand between task v_i and v_j
$\beta'(v_i, v_j)$	Maximum network latency allowed from task v_i to v_j
$\vec{\beta}(s^{m_i}, s^{m_j})$	Network latency between the server $s^{m_i} \in F_k$ and $s^{m_j} \in F_l, k \neq l$ in MFC environment
κ	The mean network latency between the cloud environment and the fog environment
P_i	Priority of a task v_i
M_i	Makespan of a task v_i
$\hat{M}_i$	Normalized makespan of a task v_i
$\hat{P}_i$	Normalized Priority of a task v_i
$\hat{R}_i$	Normalized computing resource demand of a task v_i

the proximity the network latency between user and fog server is very less compared to the network latency between user and cloud. As a result, user can experience the real-time performance of different services, for instance services such as online gaming service, online video analysis service which depends on huge data exchange between user device and cloud or FN, etc.

The relationship among the set of cloud servers, fog servers, and the users can be represented as a graph structure. The list of notations that are used through-out the paper including the problem models in Section 4 and 5 is given in Table 2.

3.1. Resource Graph

The resources of both cloud and fog servers are represented in a graph structure, known as resource graph. Let $G = (V, E)$ be the resource graph, where $V = \{C, F\}$ be the set of servers available in both Cloud and Fog environment and E be the set of edges among those servers.



(a) Resource graph consisting of one FN and one cloud environments.

(b) Resource graph consisting of two FNs and one cloud environments

Figure 2: An example of resource graph.

An example of such resource graph can be seen in Figure 2a. Further, $C = \{s_c^1, s_c^2, s_c^3, .., s_c^{n_c}\}$ be the set of servers available in cloud environment and $F = \{s_f^1, s_f^2, s_f^3, .., s_f^{n_f}\}$ be the servers available in nearby fog environment. In Figure 2a, the cloud environment consisting of five servers, $C = \{s_c^1, s_c^2, s_c^3, s_c^4, s_c^5\}$, and the fog environment consisting of three servers, $F = \{s_f^1, s_f^2, s_f^3\}$. For each server $s^{m_1} \in V$, $R^x(s^{m_1})$ represents the remaining resource available of type $x \in \{CPU, Memory\}$. Available bandwidth and network latency are the two parameters that are associated with each physical edge $e(s^{m_1}, s^{m_2})$. Let $\alpha(s^{m_1}, s^{m_2})$ and $\beta(s^{m_1}, s^{m_2})$ be the available bandwidth and network latency, respectively, on physical link between server s^{m_1} and s^{m_2}, $(s^{m_1}, s^{m_2}) \in C$ or $(s^{m_1}, s^{m_2}) \in F$. The mean network latency between the cloud and the fog environment is represented by κ. Since the network latency is assumed to be time-variant, the mean value is always considered during the placement of the tasks. Further, it is assumed that there exists a physical link between each pair of fog servers. For example the servers in Figure 2a are connected directly without any FCI in-between. Mathematically, $e(s^{m_1}, s^{m_2}) = 1, \forall (s^{m_1}, s^{m_2}) \in F$. In other words, the fog servers are connected to each other in a complete graph. The other form of connection i.e. incomplete graph fog servers is discussed in Section 5, where it is considered as each

pair of fog servers may not be connected directly, as the servers may belong to different fog environments in different geographical locations (e.g. in Figure 2b).

3.2. Task Graph

The users' requests are assumed to be comprised of multiple inter-connected tasks, represented as Directed Acyclic Graph (DAG), $G' = (V', E')$. An example of user's application is presented in Figure 1b, which consists of 6 tasks, $V1 - V6$. Let $V' = \{v_1, v_2, ..., v_{n'}\}$ be the set of tasks and E' be the set of edges among tasks. Each task v_i is associated with resource demand of type CPU and memory, denoted by $\bar{R}^x(v_i), x \in \{CPU, Memory\}$. Let $\hat{e}(v_i, v_j)$ be the boolean variable indicating if there is an edge existing from task v_i to task v_j. The direction of the edge also represents the dependencies of task v_j upon the task v_i. This dependency indicates that the dependent task v_j cannot start its execution before task v_i, as task v_j needs input from task v_i. Since the tasks graph is modelled as DAG, the value of $\hat{e}(v_i, v_j) + \hat{e}(v_j, v_i) \leq 1$.

Taking the network resource demand into account, let $\alpha'(v_i, v_j)$ be the bandwidth demand between the task v_i and v_j. In order to establish the smooth communication among the tasks, let $\beta'(v_i, v_j)$ be the maximum network latency allowed from task v_i to v_j.

4. Problem formulation

Considering the resource graph and the task graph, the main goal is to map or place the interconnected tasks onto both cloud and fog servers such that the workload distribution among the physical servers can be balanced and the network latency can be minimized. The fog servers are given higher preference over the cloud servers while mapping process. The mapping process comprises of two stages: Task mapping and task edge mapping. In task mapping stage, all the tasks are mapped onto multiple servers. Here, we exploit the opportunity to map multiple tasks from same task graph onto the same physical servers. As a result; the number of physical servers involved is less than the number of tasks present in the task graph. In the task edge mapping stage, all the edges between the tasks are mapped onto the physical edges considering the network bandwidth and the network latency.

4.1. Task-edge mapping

The boolean variable $\gamma^i_{c,j} = 1$, if the task v_i is placed onto cloud server s^j_c, 0 otherwise. Similarly, $\gamma^i_{f,k} = 1$ if the task v_i is placed onto fog server s^k_f, 0 otherwise.

The boolean variable $D^{i_1 i_2}_{m_1 m_2}$ represents if the edge between task v_{i_1} and v_{i_2} is mapped onto the physical edge among cloud and fog servers $s^{m_1} \in V$ and $s^{m_2} \in V$. When a task edge needs to be mapped onto the physical edge between one cloud server and one fog server, an additional network latency κ would be considered. In order to differentiate the set of fog servers and the cloud servers, the value of $\beta(s^{m_1}_c, s^{m_2}_f) = \kappa, \forall s^{m_1}_c \in C, \forall s^{m_2}_f \in F$ must be larger than the network latency between any two cloud or fog servers. Mathematically,

$$\kappa > \max_{0<m_1,m_2\leq n_c} \{\beta(s^{m_1}_c, s^{m_2}_c)\} \tag{1}$$

$$\kappa > \max_{0<m_1,m_2\leq n_f} \{\beta(s^{m_1}_f, s^{m_2}_f)\} \tag{2}$$

A precise summary of the assumptions considered in the context of single FN and cloud environment can be made as follows. A user's request is assumed to be comprised of multiple inter-connected tasks and hence no task is isolated from the other tasks in the same request. The fog servers within a FN are connected and there exists a physical link between each pair of fog servers. It is assumed that FCIs can communicate among themselves either over wired or wireless network connection. The network connection in the whole multiFog-cloud environment is time-variant.

4.2. Objective function

The problem of mapping the users' interconnected tasks to cloud and fog environment can be formulated as mixed integer linear programming problem. The main objectives are to balance the workload by uniform distribution of the tasks and minimize the network latency.

Hence, the main objective can be divided into two sub-objectives: mapping the tasks to the servers and virtual edges to the physical edges. The following equation represents the

assignment of a task $v_{i_1} \in V'$ onto a fog or cloud server.

$$z^{v_{i_1}} = \sum_{\forall s^{m_1} \in V} \left(\gamma_{c,m_1}^{i_1} + \Delta * \gamma_{f,m_1}^{i_1}\right) * \frac{1}{R^x(s^{m_1})} \tag{3}$$

Similarly, the mapping of a virtual edge $\hat{e}(v_{i_1}, v_{i_2}) \in E'$ onto a physical edge can be represented as follows:

$$z^{(i_1 i_2)} = \sum_{\forall (s^{m_1}, s^{m_2}) \in E} D_{m_1 m_2}^{i_1 i_2} * \hat{e}(v_{i_1}, v_{i_2}) * \left[\beta(s^{m_1}, s^{m_2}) + \frac{1}{\alpha(s^{m_1}, s^{m_2})}\right] \tag{4}$$

Using the Equation 3 for task mapping and Equation 4 for virtual edge mapping, the primary objective function for the user's request G' can be formulated as below.

Objective:

$$\min Z = \sum_{\forall v_{i_1} \in V'} z^{v_{i_1}} + \sum_{\forall (v_{i_1}, v_{i_2}) \in E'} z^{(i_1 i_2)} \tag{5}$$

Constraint:

$$\gamma_{c,j}^{i} + \gamma_{f,k}^{i} = 1, \quad \forall s_c^j \in C, \forall s_f^k \in F \tag{6}$$

$$\bar{R}^x(v_i) < R^x(s_c^j) * \gamma_{c,j}^{i} + R^x(s_f^k) * \gamma_{f,k}^{i} \tag{7}$$

$$\alpha(s^{m_1}, s^{m_2}) > \alpha'(v_{i_1}, v_{i_2}) \tag{8}$$

$$\beta(s^{m_1}, s^{m_2}) < \beta'(v_{i_1}, v_{i_2}) \tag{9}$$

$$0 < \Delta < 1; \quad 0 < i_1, i_2 \leq n_c; \quad 0 < m_1, m_2 \leq n_f \tag{10}$$

The objective function is of two folds:

- The first part of the function allows all the tasks to be mapped onto the physical servers available at nearby FN and cloud. Δ is used as a constant to encourage the tasks to be mapped in fog servers instead of cloud servers. Since, the objective is to minimize, the task will be mapped to the servers having maximum remaining resource. Further, if two servers (one at FN and other at cloud) having minimum resources, the task will be mapped to the fog server due the multiplication of Δ constant.

- The second part of the function (derived from Equation 4) minimizes network latency while mapping the virtual edges to the physical edges. $\beta(s^{m_1}, s^{m_2})$ in Equation 4 imposes additional network latency when two servers, s^{m_1} and s^{m_2}, are in different computing environments. As a result, this discourages using any physical connection between cloud and fog as this would give a larger value of β. Minimizing the term $\frac{1}{\alpha(s^{m_1}, s^{m_2})}$ in Equation 4 infers that the physical edges with maximum bandwidth available are highly preferred for any virtual edge.

However, the solution must meet the following constraints.

- Constraint (6) ensures that all the tasks are mapped onto exactly one server. The physical server must either be cloud server or fog server.

- Constraint (7) and (8), ensure that the tasks and the edges between the tasks are mapped onto the physical servers and physical edges with enough remain resources available.

- Similarly, Constraint (9) ensures that the required network latency is fulfilled by the physical edge. On the other hand, Constraint (10) ensures that the value of Δ must lies between 1 and 0 to encourage the tasks to be mapped onto the fog server.

5. MultiFog-Cloud (MFC) Environment

In Section 4, the system consists of one cloud and one FN. For each user, the fog servers are connected to each other forming a complete graph. However, in order to fit into the real-scenario, a MultiFog-Cloud (MFC) environment is considered, as shown in Figure 1a, where more than one FNs are deployed in different geographical locations. All the FNs are directly or indirectly connected to each other. As a result, the fog servers in all FNs are forming an incomplete graph and no fog server is unreachable from any other fog server.

5.1. Multi-Fog resource graph

As the system model, in Section 3, is updated, the resource graph can be represented as $V = \{C, F_1, F_2, ...\}$. For example, in Figure 2b, two fog environments: f_1 consisting of

three servers and f_2 consisting of two servers are connected to the cloud environment with five servers. Each fog consists of multiple servers represented as $F_i = \{s^1_{f_i}, s^2_{f_i}, s^3_{f_i}, ...\}$. Each server is represented as $s^j_{f_i}$, which indicates server j in FN F_i. $|F_i|$ represents the number of servers present in FN F_i. It is assumed that the number of servers in all FNs are not uniform. In order to get the location of a server s^{m_j}, the notation $L(s^{m_j})$ is used. As the resource graph is not complete, the term physical edge is different from physical path. Physical edge can be defined as the direct link between two servers. On the other hand, a physical path is the combination of multiple physical links connected to each other to transfer data from one end to other ends. A physical link can be between FN and FCI, two FCIs, between FCI and cloud. The term physical path is used when establishing a connection between two servers from different FNs. The notation $\vec{P}_{ij}$ is used to indicate the physical path between server $s^{m_i} \in F_k$ and $s^{m_j} \in F_l, k \neq l$. If both the servers belongs to same node, the path $\vec{P}_{ij}$ is same to the notation $e(s^{m_i}, s^{m_j})$. $\vec{H}_{ij}$ indicates the number of hops present in between the server $s^{m_i} \in F_k$ and server $s^{m_j} \in F_l, k \neq l$. E represents the set of all physical edges and physical paths present in the multi-fog and cloud environment. $\vec{\beta}(s^{m_i}, s^{m_j})$ represents the network latency between the server $s^{m_i} \in F_k$ and $s^{m_j} \in F_l, k \neq l$. It is assumed that network latency of any physical link is smaller than the network latency of any path. Mathematically,

$$\begin{aligned} \max\{\beta(s^{m_i}, s^{m_j}) | \forall F_k, s^{m_i} \text{and} s^{m_j} \in F_k\} < \\ \min\{\vec{\beta}(s^{m_i}, s^{m_j}) | \forall F_k, F_j, s^{m_i} \in F_k, s^{m_j} \in F_l\} \end{aligned} \tag{11}$$

There is no relationship between network latency of a path between FN and cloud and the network latency of fog paths.

5.2. Resource mapping in multi-fog environment

Considering the new multi-fog cloud model, the complexity of resource mapping problem leverages when the mapping scope stretched from single-fog single-cloud scenario to multi-fog single-cloud scenario. The user's tasks can be mapped onto nearby fog, remote fog or remote cloud computing environment. During mapping, multiple parameters need to be

taken into account, such as the number of hops, network latency, workload distribution, etc. The boolean variable $\gamma^i_{f_j,k} = 1$, if the task is placed onto the server k in the FN (except the nearby FN) F_j, 0 otherwise. However, the boolean variable $\hat{\gamma}^i_{f_j,k} = 1$, if task v_i is placed onto a nearby fog server $s^k_{f_j}$, 0 otherwise.

The task edges can be mapped onto the physical edge of physical path. In order to achieve the goal of minimizing network latency, the physical edges are given higher preferences over physical path while mapping the task edge. The boolean variable $D^{ij}_{m_k m_l}$ represents if the task edge between task v_i and v_j is mapped onto the physical edge $e(s^{m_k}, s^{m_l})$ or the physical path $\vec{P}_{kl}$.

A precise summary of the key assumptions considered in the context of multi-fog and cloud environment (in addition to the key assumptions in Section 3) can be made as follows. The resource configurations/capacities in all FNs are heterogeneous. It is also assumed that the network latency of any physical link is smaller than the network latency of any path. Hence, the latency value between any FN and the corresponding FCI is smaller than the latency value among FCIs and between FCIs and the cloud. The proposed work assumed that the user assigns a priority to each task. When tasks are being scheduled, the effect of resource availability fluctuation is negligible.

5.3. Objective function

Extending the objective function in Section 4.2, the problem of mapping the user's task graph onto multi-fog and cloud environment can be formulated as mixed integer linear programming problem with the goal to minimize the total network latency and distribute the workload among fog and cloud servers in a balanced manner. The tasks are mapped in such a way that total network latency among tasks is minimum without violating the computing resource demand.

From Equation 5, the mathematical formulation of single task $v_{i_1} \in V'$ mapping onto a

physical server available at multi-fog single-cloud environment can be represented as

$$\hat{z}^{v_{i_1}} = \sum_{\forall s^{m_1} \in V} \frac{\left(\gamma^{i_1}_{c,m_1} + \gamma^{i_1}_{f_j,m_1} + \Delta * \hat{\gamma}^{i_1}_{f_k,m_1}\right)}{R^x(s^{m_1})} \tag{12}$$

For further simplification, mapping virtual to physical edges is associated with two objectives: minimizing the network latency and balancing the network workload. While mapping a virtual edge, (v_{i_1}, v_{i_2}), to a physical edge, the network latency can be calculated as below.

$$\hat{z}^{(i_1 i_2)}_{latency} = \sum_{\forall (s^{m_1}, s^{m_2}) \in E} \Big\{ \hat{e}(v_{i_1}, v_{i_2}) * \big(D^{i_1 i_2}_{m_1 m_2} * \beta(s^{m_1}, s^{m_2}) + D^{i_1 i_2}_{m_1 m_2} * \vec{\beta}(s^{m_1}, s^{m_2}) + \vec{H}_{s^{m_1}, s^{m_2}} \big) \Big\} \tag{13}$$

Similarly, the mathematical formulation for mapping a virtual edge, (v_{i_1}, v_{i_2}), onto a physical edge taking bandwidth availability into account can be represented as,

$$\hat{z}^{(i_1 i_2)}_{bandwidth} = \sum_{\forall (s^{m_1}, s^{m_2}) \in E} \frac{D^{i_1 i_2}_{m_1 m_2} * \hat{e}(v_{i_1}, v_{i_2})}{\alpha(s^{m_1}, s^{m_2})} \tag{14}$$

Considering the Equations 12, 13, and 14, the previous objective function in Equation 5 can further be modified as follows.

Objective:

$$\min \hat{Z} = \sum_{\forall v_{i_1} \in V'} \hat{z}^{v_{i_1}} + \sum_{\forall (v_{i_1}, v_{i_2}) \in E'} \left[\hat{z}^{(i_1 i_2)}_{latency} + \hat{z}^{(i_1 i_2)}_{bandwidth} \right] \tag{15}$$

Constraint:

$$\gamma^{i}_{f_j,m_1} + \gamma^{i}_{f_k,m_1} \leq 1 \tag{16}$$

$$\beta'(v_1, v_2) \geq \vec{\beta}(s^{m_1}, s^{m_2}) \tag{17}$$

$$\sum_{\forall v_i \in V'} \hat{\gamma}^{i}_{f_j,k} \geq 1 \tag{18}$$

In the first part of the function, servers available at nearby FN, other FNs, and the cloud environments are considered while assigning a task. The second part of the objective function focuses on mapping the virtual edges onto the physical edges. If no suitable physical edge is available, a physical path consisting of multiple physical edges will be chosen (details in Section 6). The number of hops and the network latency are collectively considered while selecting a physical path. In addition to the constraints mentioned in Equations 6-10, the objective function in Equation 15 must satisfy following constraints.

- Constraint 16 ensures that no task is mapped onto multiple FNs. On the other hand, Constraint 17 ensures that the maximum allowed network latency of any task edge $\hat{e}(v_{i_1}, v_{i_2})$ is smaller than the corresponding physical path $\vec{P}_{m_1 m_2}$.
- Constraint 18 ensure that at least one task is placed in the nearby FN. This would allow the service provider to deploy the most critical task in the task graph onto the nearby FN. This would also prohibit the service provider to place the entire task graph in a very remote FN

6. Proposed Algorithm

In this section, we discuss the proposed novel Heuristic Resource Allocation and optimization in MultiFog-Cloud (HeRAFC) algorithm, which works in a non-distributed manner and the users requests are handled on FCFS manner. Figure 1a shows the architecture of multiple fogs and cloud and Figure 1b shows an example of users' application. As discussed in earlier section, the job of this algorithm is to distribute the application tasks among nearby FN, the FNs at multi-hop distance and the cloud. Each FN consists of multiple interconnected fog servers and network devices such as switches, routers, etc.

Definition 1. 1-hop distance: A FN F_i is said to be at 1-hop distance from other FNs $\forall F_j, F_i \neq F_j$, if both the FNs, F_i and F_j, share same FCI. Taking Figure 1a into account, for all the servers located in cloud C, all the FNs in $FCI1, FCI2$, and $FCI3$ are at 1-hop distance, but not the FNs that are connected to $FCI4$. Similarly, FN F_1 and F_2 under $FCI1$ are at 1-hop distance from each other.

Definition 2. 2-hop distance: A FN F_i is said to be at 2-hop distance from other FNs $\forall F_j, F_i \neq F_j$, if the corresponding FCIs can communicate directly with each other without cloud. For example, fog F_1 under $FCI1$ is at 2-hop distance from all the FNs that are connected to $FCI2$. Additionally, the servers in cloud C and the FNs connected to $FCI4$ are at 2-hop distance from each other.

Definition 3. n-hop distance: In general, two FNs are said to at n-hop distance, if $n, n > 0$ number of FCIs are involved in the physical path between corresponding FNs. For example in Figure 1a, the FN under $FCI4$ is at 3-hop (here $n = 3$) distance from the FN F_2 under $FCI1$.

Three major parameters associated with the users' applications are taken into account: makespan of the task, priority value of each task, and computing resource demand that includes CPU and memory demand.

Definition 4. Task makespan: Makespan of a task v_i, M_i refers to the time required to execute the task, also referred to as the task's execution time. This excluded the response time. The makespan of the tasks can be used to derive the makespan of the application.

Definition 5. Priority value of task: Priority of a task v_i, P_i, refers to how important the task is. The value of P_i is decided by the user. Higher the value of priority, more important the task is. Mathematically, a task v_1 is said to be higher priority than the task v_2, if $P_1 > P_2$.

Based on above-mentioned three major parameters, the critical value of each task is calculated. This work considers parameters with a wide range of values. For example, makespan of a task is in time unit and the value may range from 1 to several thousands. Similarly, memory resource demand value is in megabytes and the value may range from hundreds to thousands. The value of CPU resource demand may reach up to 100 and the priority of tasks can be of any discrete integer value. For such a diverse range of values, it is necessary to normalize the values to a range of (0, 1], i.e. greater than 0 and less than or equal to 1. Without normalization, a slight increase in one parameter may have a

substantial impact on the final result. On the other hand, a significant change in another parameter may result in a slight change in the final result. The normalized makespan of a task is calculated as:

$$\hat{M}_i = \frac{M_i}{\max\{M_i | \forall v_i \in V\}} \tag{19}$$

Similarly the normalized priority value of a task can be calculated as

$$\hat{P}_i = \frac{P_i}{\max\{P_i | \forall v_i \in V\}} \tag{20}$$

Since computing resource refers to the CPU and memory demand which are in different units, the normalized value of each type of resource demand is calculated followed by calculating the average of both normalized values. Mathematically, the normalized CPU demand can be calculated as:

$$\hat{R}^{CPU}(v_i) = \frac{\bar{R}^{CPU}(v_i)}{\max\{R^{CPU}(s^{m_1}) | \forall m_1 \in \{C, F_1, F_2, \dots\}\}} \tag{21}$$

The normalized value of memory demand can be calculated as:

$$\hat{R}^{mem}(v_i) = \frac{\bar{R}^{mem}(v_i)}{\max\{R^{mem}(s^{m_1}) | \forall m_1 \in \{C, F_1, F_2, \dots\}\}} \tag{22}$$

Taking the value calculated using Equation 21 and 22, the normalized average weighted computing resource demand can be calculated as follows.

$$\hat{R}_i = \frac{\Omega^c * \hat{R}^{CPU}(v_i) + \Omega^m \hat{R}^{mem}(v_i)}{2} \tag{23}$$

, where Ω^c and Ω^m are the constants, $\Omega^c + \Omega^m = 1, 0 < \Omega^c, \Omega^m < 1$. These constants are used to assign preference values to CPU and mem resource.

Considering the normalized values calculated in Equation 19-23, the critical value of a task is calculated. Weighted Multi-Dimensional (WMD) approach is followed, where each parameter represents one dimension in a multi-dimensional space. In this context, the

makespan, priority value and computing resource demand represent three dimensions of a space. In this 3-Dimentional (3D) space, a task is represented as a 3D rectangular prism object. The volume of such 3D rectangular prism object represents the critical value of that corresponding task. However, in order to allow the service provider to give higher preference to one parameter over other, a weight factor is used in the critical value calculation. The critical value (or the volume of the 3D object) of a task v_i can be calculated as

$$WV(v_i) = (w_1 * \hat{M}_i) * (w_2 * \hat{P}_i) * (w_3 * \hat{R}_i), \quad w_1 + w_2 + w_3 = 1 \tag{24}$$

6.1. HeRAFC algorithm

The whole process of the proposed resource allocation strategy can be divided into two phases. In the first phase, the order of tasks, based on WMD approach, for deployment is decided. In the second phase, suitable locations for each task is chosen.

6.1.1. Order of task

The WMD-based algorithm for deciding the order of tasks is presented in Algorithm 1. The user's application is taken as the input to this algorithm. The job of this algorithm is to arrange the tasks in queue, which further would be followed to assign or map to the fog-cloud environment. The detail description of this algorithm is presented in Section 6.2.

6.1.2. Location selection

In the second phase of the HeRAFC algorithm (as persented in Algorithm 2), the ordered task queue is followed and for each task, a suitable location is searched, known as node mapping. Location here refers to one FN or the cloud. Following this, all the edges among tasks are mapped by finding the suitable path from the corresponding source to destination environment where both the end tasks are already mapped. The detail description of this algorithm is presented in Section 6.2.

6.2. HeRAFC description

User's application and the information regarding the physical environment are provided to Algorithm 2 as the input. The algorithm starts by invoking Algorithm 1 that would decide

Algorithm 1: WMD-based Task order selection algorithm

Input: User's application; MFC environment

1 ST ={Sort the list of tasks in ascending order based on the number of out-edges};
2 $ProcessQ$ ={ } /* *Order of tasks need to be deployed* */ ;
3 TL ={Extract the list of tasks from ST with number of out-edges 0};
4 **while** *True* **do**
5 **foreach** *task* $t \in TL$ **do**
6 Calculate critical value $WV(t)$ using Equation 24;
7 Calculate number of out-edges $OD(t)$;
8 $MCV(t) = WV(t)/(OD(t) + \delta)$ /**Calculate mean critical value of the task*/ ;
9 **end**
10 Sort TL based on their MCV in ascending order;
11 Append TL to $ProcessQ$;
12 Append -1 to $ProcessQ$ /* *to mark the tasks in level* */ ;
13 tmp = {Extract the list of parent tasks of TL from ST};
14 Remove all the tasks in TL;
15 Add the tasks in tmp to TL;
16 **end**

the order of the tasks that need to be followed in the location selection phase. Algorithm 1 starts by sorting the list of tasks based on the number of child tasks. To calculate the number of child tasks of a parent task, the number of out-edges is calculated as in Line 1. Out-edge refers to the number of outgoing edges from a task. From the sorted task list, the tasks that are having out-edge value 0 or the task having no child are first selected, as in Line 3. This means, the algorithm gives higher priority to leaf tasks while processing. This way, this algorithm takes the tasks' dependency into consideration.

For each extracted task,the mean critical value (which eventually follows WMD approach) is calculated, in Line 5-8, taking the ratio of critical value to the number of out-edges. For the leaf node the number of out edges is 0. To avoid any *Divide by Zero error*, a small constant δ is added to $OD(t)$. The extracted list is further sorted based on the mean critical value and appended to the process task queue, in Line 10 and 11, respectively. The same process is applied to the parent task of the currently extracted tasks, as in Line 15. Processing the tasks level-by-level is taken into account to ensure that the dependent tasks (or the child tasks) are executed not before the execution of their predecessor tasks (or the parent tasks). The time complexity of this algorithm is $\mathcal{O}(|V'| \log |V'|)$, where V' be the

Algorithm 2: Resource Allocation in MultiFog-Cloud algorithm

Input: User's application; MFC environment

ProcessQ =Queue of tasks decided by Algorithm 1;
Calculated the resource availability matrix RM;
$mapQ = \{\}$;
while *True* **do**
    ETL ={Extract the list of tasks from *ProcessQ* until -1 in LIFO manner.};
    foreach *(task t in ETL) AND (t is not scheduled)* **do**
        CFL ={get the list of fogs or cloud where the child tasks are deployed};
        $nf = CFL$;
        if $CFL == NULL$ **then**
            nf =Nearby FN of user's application;
        end
        Check if t can be deployed on any one of the fog or cloud in nf;
        if t *deployed on* nf **then**
            Append t to *mapQ* and update RM ;
            Continue with next task from Step 6 ;
        end
        oh ={Find the FNs or cloud that are one hop distance to the all fogs in nf};
        Check if t can be deployed on any one of the fog or cloud in oh;
        if t *deployed on* oh **then**
            Append t to *mapQ* and update RM ;
            Continue with next task from Step 6 ;
        end
        th ={Find the FNs or cloud that are two hop distance to the all fogs in nf} ;
        Check if t can be deployed on any one of the fog or cloud in th ;
        if t *deployed on* th **then**
            Append t to *mapQ* and update RM;
            Continue with next task from Step 6 ;
        end
    end
    E' ={set of task edges that are connected to the tasks in ETL};
    Sort E' in descending order based on bandwidth demand;
    foreach *edge* $e \in E'$ **do**
        Find source and destination task, s & t;
        sh = Find the FN or cloud where s is deployed;
        th = Find the FN or cloud where t is deployed;
        if $sh \neq NULL$ *OR* $th \neq NULL$ **then**
            Find the shortest path from sh to th and map the task edge;
        else
            Ignore the current edge e;
        end
    end
    Reset RM and continue with next set of tasks.;
end

set of tasks (proof in Appendix Appendix A.1).

The output of Algorithm 1, $ProcessQ$, is further followed to map the tasks and the edges. Each set of tasks are separated by -1 in $ProcessQ$, as in Algorithm 1, Line 12, can be logically represents as the tasks in one level of the task graph. The set of tasks, ETL, is first extracted from the $ProcessQ$, as in Line 5 in a Last-In-First-Out (LIFO) manner. Extracting in LIFO would further ensure that the dependent tasks (or child tasks) are not executed before the execution of predecessor tasks. This indicates that the root task in the DAG will be the first one to get executed. For each task in ETL, the set of child tasks is calculated followed by forming a list of FNs and cloud, CFL, that are hosting those child tasks, in Line 7. The set of hosting environment, CFL is considered as nearby fog list, nf, in Line 8. However, in case of absence of no child task, the FN that is nearer to the user is considered, as in Line 10. In Line 12 - 15, the current task is checked if this can be assigned to any one of the nearby FNs list. If the amount of resource can be fulfilled by the anyone of the FN or by the cloud, the task will be assigned to the selected FN or cloud and the same process will be applied to next task. However, if no hosting environment is found in Line 12, the list of FNs and cloud that are at one hop distance, oh from the child task is calculated, as in Line 17. On calculation of the set oh, in Line 18 - 21, all the hosting environments will be checked if the current task can be deployed. In case of availability of the resources on any FN that can fulfill the demand, the current task will be assigned to the selected FN or the cloud. Further, in case of failure of resource availability for the current task, the hosting environments that are at two hops distance will be calculated in Line 23 and the same process will be applied for checking the resource and assigning the task, as mentioned in Line 24 - 27. This concludes the procedure for mapping the tasks on to different FNs and cloud.

Upon mapping the set of tasks, ETL, the set of adjacent edges, E' in Line 30, will be assigned, in Line 32 - 39. For each edge, e, the information regarding the source task and the destination task are extracted from the task graph, in Line 33. If both the tasks are already mapped onto the physical environment, the shortest path between the corresponding hosting environment will be calculated and assigned to host the edge e, in Line 37. Upon

mapping of both the set of tasks in one level of task graph and the edges among them, the resource matrix will be reset, in Line 42. This is due to the fact that one level tasks can not run in parallel to the other level of tasks (parent or child tasks) and need to be scheduled to run in a sequential manner. Hence, upon completion of execution of one level of tasks, the leased resources will be released for the set of tasks in the next level of the task graph. The time complexity of the proposed algorithm is $\mathcal{O}(|E'|\,[\log|E'| + |E|log(|V|)])$, where E and E' are the set of physical edges and task edges, respectively and V is the number of cloud and FNs (proof in Appendix A.2).

7. Performance evaluation

In this section, the performance of the proposed algorithm is discussed. Liu *et al.* (DRACO) [4] and Shah-Mansour *et al.* (HFCCI) [5] algorithms are used to compare and present the efficiency in terms of computing and network resource utilization in different hosting environment, such as fog and cloud. FNs nearby can receive data from multiple users in case of DRACO algorithm. By using game theory, HFCCI minimizes the number of users and reduces energy costs while allocating cloud resources. Various performance matrices are employed for evaluating efficiency, including the effect of tasks' order on resource utilization, a comparison of network resource utilization between fog and cloud environments, and tasks' latency. The simulation uses a rule-based approach over a data-driven ML-based approach. Rule-based approaches can be used to generate simulated environments that mimic the real-world. This uses predefined configurations and rules, and does not require any historical data to fit the environment. On the other hand, a data-driven ML-based approach needs historical data that fits the environment and the problem statement. In the context of heterogeneous fog and cloud resource allocation and optimization, the data related to historical performance of the computing and storage cluster, network resource utilization, real requests from the users are difficult to acquire. To the best of our knowledge, there are no such data available that fit the problem addressed in this paper. This is the main reason why most of the research community and we preferred a rule-based approach over a data-driven ML-based approach. In the following subsection, the detail

information on simulation environment and the results obtained from the simulation are presented.

7.1. Simulation environment

A Python-based discrete event simulator is used to simulate the proposed resource management algorithm. This simulation does not explicitly configure tasks to run on different CPU cores. This is entirely dependent on the number of cores available and how the native operating system behaves. The resource configuration of the FCIs, fog, cloud and the users' applications are presented in JSON file. The applications are created dynamically/programmatically. Separate JSON files are created for each user's app, including the tasks list, edges list, resource demand of tasks and edges, and other related information. The detailed configuration of the simulation parameters are given in Table 3. Appendix B, provides a statistical analysis of the entire environment consisting of users' applications, tasks, resource demand, FNs, FCIs, resource availability and priority, connectivity among FNs, FCI and cloud, and network resource availability.

The MFC environment consists of 500 FNs that are connected to 200 FCIs. Each FCI is connected to at least one FN, whereas a single FN is connected to precisely one FCI. For each FN, the total amount of CPU capacity ranges from 50 through 100 and the values are assigned by following a random distribution. The total memory (RAM) for each FN is assigned randomly ranging from $200GB$ to $400GB$, where the unit GB represents GigaByte. The approximate computation power of a FN is calculated as MIPS (Million instructions per second), which ranges from 3000 to 5000.

As discussed before, each FN is connected to exactly one FCI. The maximum network bandwidth available between a FN and the corresponding FCI ranges from $300Mbps$ through $400Mbps$. In this paper, it is considered that a FCI may directly communicate with another FCI. In such a situation, there exist a network bandwidth capacity among FCIs, which ranges from $400Mbps$ through $1000Mbps$. Similarly, the bandwidth between FCIs and the cloud range from $400Mbps$ through $1000Mbps$. The network bandwidth unit is in Mbps (Mega bits Per Second). It is assumed that latency value between any FN and the corre-

sponding FCI is small than the latency among FCIs and between FCIs and cloud. Latency values are assigned to the physical links by following a uniform distribution. The latency (in millisecond) between FN and FCIs ranges from $50ms$ through $100ms$. Similarly, the latency values for rest of the physical links range from $100ms$ through $200ms$. Technically, a task can be hosted on 1-hop or at a multi-hop distance. However, in the current implementation maximum number of hops is set to 2. This means there are a maximum of two FCIs and no cloud are present in the physical path between any two FNs that are used to host user's tasks.

The maximum number of applications is set to 10000. For each application, random number of tasks are generated ranging from 4 through 12. However, the total maximum number of tasks for the entire simulation is set to $100,000$. For each task, the CPU and memory demand range from $1 - 4CPUs$ and $100MB - 1000MB$, respectively. The average makespan of each task is $350ms$ with minimum and maximum makespan of $10ms$ and $1000ms$, respectively. The network bandwidth demand of edges is ranging from $100Mbps$ and $200Mbps$. Similarly, the average latency demand among tasks ranges between $10ms$ and $50ms$. The unit of latency is millisecond (ms). The minimum and maximum priority values of a task are 1 and 5, respectively. The bandwidth and latency demand of a task ranges between $100 - 200Mbps$ and $10 - 300ms$, respectively. The number of edges within an application is decided by a link probability of 0.6. This indicates the connectivity probability among two task.

7.2. Simulation results

Following the discussion of simulation environment in above subsection, this subsection presents the results of the simulation in more details. Figure 3 and Figure 4 provide the resource utilization of fog and cloud environment, respectively. In this simulation implementation, resource refers to CPU, Memory and network bandwidth. Figure 3 shows the comparison of resource utilization of proposed HeRAFC algorithm with DRACO [4] and HFCCI [5] algorithms only in fog environment. Resource utilization is calculated as the ratio of amount of resources allocated and the total resource capacity.

Table 3: Simulation parameters for Fog-Cloud environment and users application

Simulation Parameters			
Fog-Cloud Environment		**Users Application**	
Environment configuration file format	JSON	Application configuration file format	JSON
Total number of FNs	500	Number of applications	5000-10000
Number of FCIs	200	Number of tasks per application	4-12
		Maximum number of tasks in the simulation	100,000
CPU capacity per FN	50-100	Required vCPU per task	1-4
RAM (in GB) capacity per FN	100-200	Required RAM (in MB) per task	100-1000
Average MIPS per FN	250-400	Makespan of each task (in ms)	10-1000
Bandwidth between of FN and FCI	300-400 Mbps	Priority of each task	1-5
Bandwidth between of FCI and cloud	400-1000 Mbps	Edge bandwidth demand (in Mbps)	100-200
Bandwidth among FCIs	400-1000 Mbps	Edge latency demand (in ms)	10-50
Latency between FN and FCI	50-100 ms		
Latency between FCI and Cloud	101-200 ms		
Latency among FCIs	101-200 ms		
Maximum number of hops	2		

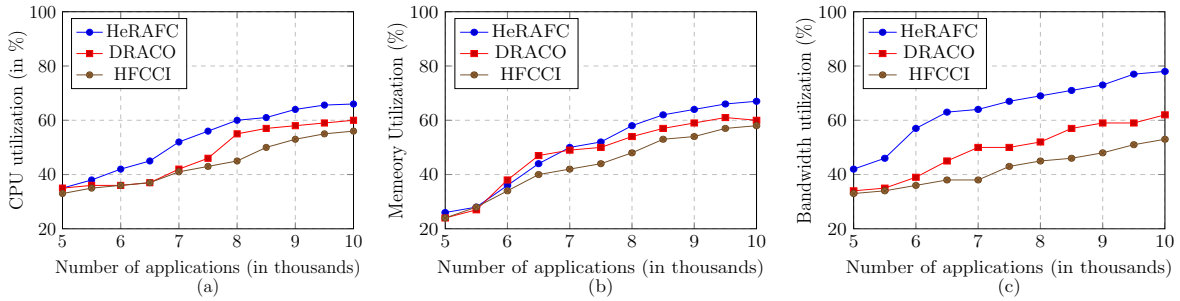


Figure 3: Comparison of only FOG resource utilization (a) CPU (b) memory (c) network bandwidth.

Figures 3(a), 3(b), and 3(c) show the average CPU, memory, and network bandwidth utilization of FNs, respectively. X-axis represents the total number of applications ranging from 5000 to 10000 and Y-axis represents the average resource utilization in percentage. It can be observed that, Upon assigning 5000 applications using the HeRAFC algorithm, the average CPU utilization of fog servers is approximately 35%, similar to the DRACO algorithm. However, it is observed, the CPU utilization is approximately 33% in the case of the HFCCI algorithm, as shown in Figure 3(a). Resource utilization continuously increases when the number of applications increases to 8000. However, when the number of applications increases beyond 8000, the rate of increase slows down. It is observed that the average CPU utilization is 66%, 60%, and 56% in case of HeRAFC, DRACO, and HFCCI algorithms, respectively, when there are a total of 10 thousand applications. A similar pattern is also observed in memory resource utilization. In case of the proposed HeRAFC algorithm the

memory utilization of the fog environment increases from approximately 26% to 67%, when the number of applications increases to 5000 to 10000. Figure 3(b) demonstrates that the proposed HeRAFC algorithm outperforms DRACO and HFCCI algorithms. The memory utilization of FNs is 60% and 58% when DRACO and HFCCI algorithms are applied, respectively, when the number of applications is 10 thousand. Overall, the proposed HeRAFC algorithm performs better than the others in utilizing the CPU and memory resources. This is due to the underlined multi-fog environment where schedulers can take advantage of the fog resources and hence assign a maximum number of tasks to the fog environment instead of the cloud. However, DRACO and HFCCI do not take advantage of the nearby available fog resources.

Figure 3(c), shows the average network bandwidth utilization of different algorithms with the number of applications ranging from 10 to 100 within fog environment. As the number of applications increases, the bandwidth utilization of all three methods also increases, but each method has a different rate of increase. It is evident from the chart that HeRAFC and DRACO have similar bandwidth utilization rates, while HFCCI has a lower bandwidth utilization rate. In case of HeRAFC, the average network utilization ranges from 42% to 78% when the number of applications ranges from 5 to 10 thousand, which is more than the network utilization obtained using other two related algorithms. This is due to the fact that HeRAFC explores the other FNs that are at multi-hop distance to fulfill the resource demand of application. As a result, the workload on the fog servers and the network among FNs are relatively higher. The average network utilization within FNs in case of DRACO and HFCCI are approximately 62% and 53%, respectively, when the number of applications is 10 thousand. It is to be noted that out of all the applications, some of the applications are hosted on FNs and rest are hosted on cloud environment. The resource utilization of the cloud environment is provided in Figure 4.

The cloud resource utilization in the case of all three algorithms is presented in Figure 4. Using HeRAFC, the average cloud CPU utilization increases from approximately 28% to 59% when the number of applications increases from 5000 to 10000, as in Figure 4(a). However, in the case of DRACO and HFCCI algorithms, the average CPU utilization in-

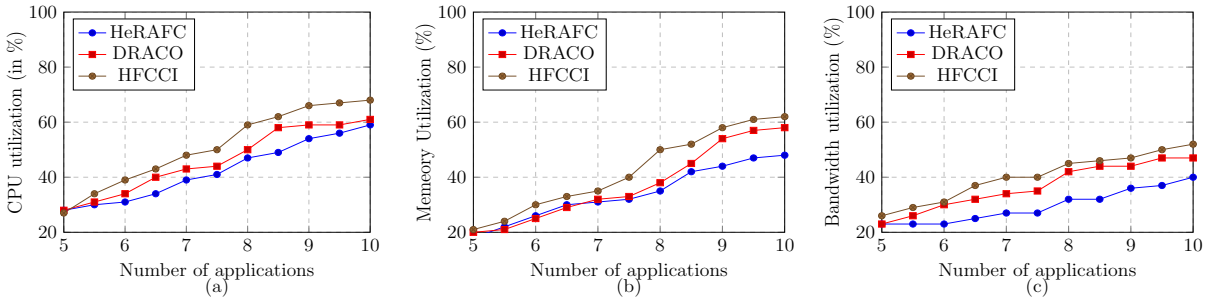


Figure 4: Comparison of only CLOUD resource utilization (a) CPU (b) memory (c) network bandwidth.

creases from 28% to 61% and from approx. 27% to 68%, respectively, with the same number of applications. Evidently, the cloud CPU utilization using HeRAFC is less than that of the other two algorithms, as the FNs (including those at a multi-hop distance) are given higher priority over the cloud while assigning the applications. A similar trend can be seen in Figure 4(b). The average memory utilization in the cloud environment using the HeRAFC algorithm with 5000 applications is approximately 18%. In contrast, the memory utilization values in the case of DRACO and HFCCI algorithms are approximately 20% and 21%, respectively. Such a utilization pattern indicates a lesser dependency on the cloud. This is because HeRAFC gives higher priority to the nearby and other FNs, letting fewer tasks be offloaded to the cloud. The impact of such a strategy can also be seen in the significant reduction of cloud network resource utilization, as discussed below.

The average network bandwidth utilization, as shown in Figure 4(c), refers to the utilization of physical links that are attached to the cloud. When the number of applications increases from 5000 to 10000, the average network bandwidth utilization using the HeRAFC algorithm increases from 23% to approximately 40%, which is less than that of the other two algorithms. For DRACO and HFCCI, the average network bandwidth utilization increased to 47% and 52% when the number of applications is increased to 10000. The HeRAFC algorithm gives higher preferences to the nearby FNs. As a result, the maximum number of task edges are assigned to the physical path within FNs, leading to relatively lesser bandwidth utilization in the cloud environment.

The results in Figure 5 and 6 show the effect of weighted critical value (WV, as calculated in Equation 24) and tasks' priority on average network latency in Fog and in cloud environments, respectively. In other words, we observed the average latency of tasks with different priority and WV values. The value of WV depends upon three parameters: makespan of the tasks (Equation 19), priority (Equation 20), and resource demand (Equation 23). By changing the value of w_1 (weight for makespan), w_2 (weight for priority), and w_3 (weight for resource demand), the value of WV can be changed. Four configurations of w_1, w_2,andw_3 are considered in the simulation. Under the first configuration, the same weightage values ($w_1 = w_2 = w_3$) are given to all the parameters (Figure 5(a)). Under the second configuration, a higher weightage is given to the resource demand of the tasks ($w_1 = 0.2, w_2 = 0.3, w_3 = 0.5$), as shown in Figure 5(b). In the third and forth configurations, higher importance is given to task's priority (Figure 5(c)) and makespan (5(d)), respectively. The latency value of a task is calculated by taking the average latency of out-going adjacent edges. So in case of an edge from a task with Priority 1 ($P_i = 1$) to a task with priority 2 ($P_i = 2$), the value of $P_i = 1$ task is calculated by considering the value of latency of edge. In other words, the latency of an edge is assigned to the source task and not the destination task. In this simulation, the tasks with priority value 5 are given higher priority and the minimum priority value of a task is set to 1. The priority values are assigned to the task following a random distribution. Figure 5 shows the latency of different priority tasks within the fog environment. The x-axis represents the total number of tasks, ranging from 50 thousand to 100 thousand and Y-axis represents average latency of different tasks in milliseconds (ms). It is observed that the average latency of the high priority tasks is always less than that of the tasks with low priority value, under different configurations.

Under the first configuration, the average latency of tasks with $P_i = 5$ and $P_i = 1$ in only fog environment is approximately $11ms$ and $93ms$ when the number of tasks is 50 thousand. The latency of those tasks increases to approx. $34ms$ and $130ms$ when the number of tasks increases to 10000, as shown in Figure 5(a). A similar pattern is observed under the forth configuration, where a higher weightage is given to the makespan parameter ($w_1 = 0.5, w_2 = 0.3, w_3 = 0.2$). The average latency of the tasks with priority ($P_i = 5$) is ap-

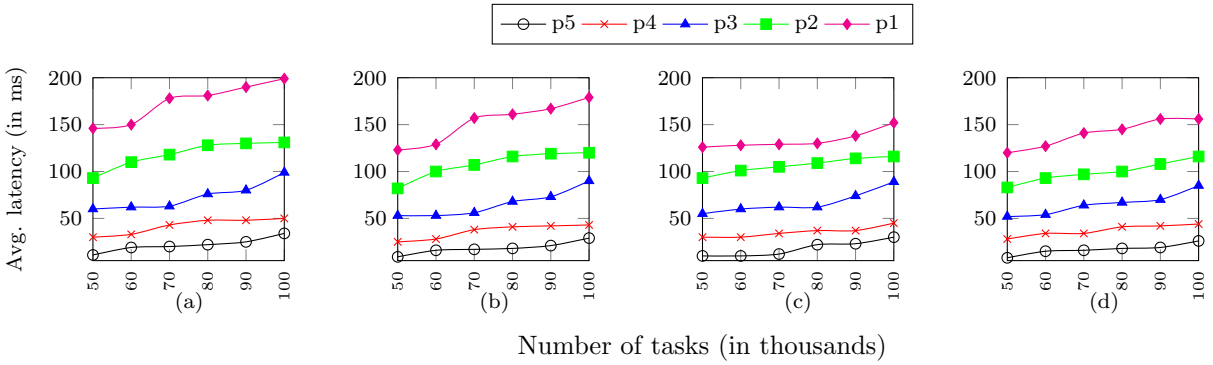


Figure 5: Latency of different priority tasks in Fog environment under different weighted critical values (WV) (a) $w_1 = w_2 = w_3$, (b) $w_1 = 0.2, w_2 = 0.3, w_3 = 0.5$, (c) $w_1 = 0.2, w_2 = 0.5, w_3 = 0.3$, (d) $w_1 = 0.5, w_2 = 0.3, w_3 = 0.2$

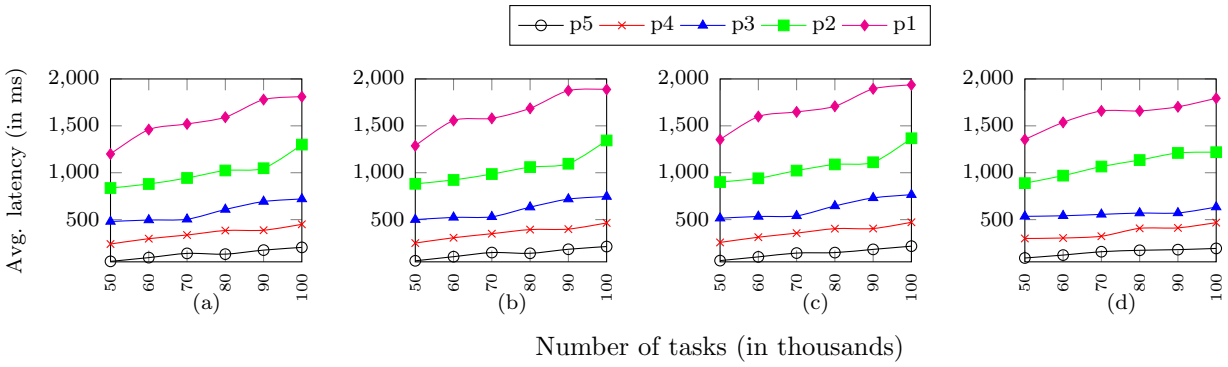


Figure 6: Latency of different priority tasks in Cloud environment under different weighted critical values (WV) (a) $w_1 = w_2 = w_3$, (b) $w_1 = 0.2, w_2 = 0.3, w_3 = 0.5$, (c) $w_1 = 0.2, w_2 = 0.5, w_3 = 0.3$, (d) $w_1 = 0.5, w_2 = 0.3, w_3 = 0.2$

prox. $8ms$, whereas the latency of $P_{=}1$ tasks is approx. $120ms$ when the number of tasks is 50 thousand. From all the observations (Figure 5(a) - 5(d)), we can conclude that under different configurations, the proposed HeRAFC algorithm gives a higher preference to the fog environment while fulfilling the resource demand and other objectives.

Unlike Figure 5, Figure 6 shows the effect of weighted critical value (WV) and the priority of tasks upon the average latency in the cloud environment. Figure 6(a) represents the average latency when all the parameters are given equal weightage, i.e., $w_1 = w_2 = w_3$. Figure 6(b), 6(c), and 6(d) represent the average latency when a higher weightage is given to resource demand, priority, and makespan parameters, respectively. Similar to the results in Figure 5, it is observed that the tasks with higher priority have less latency. For instance, as shown in Figure 6(a), the tasks with $P_i = 5$ and that are sent to the cloud environment have an average latency of $55ms$ when the number of tasks is 50 thousand. The latency increases

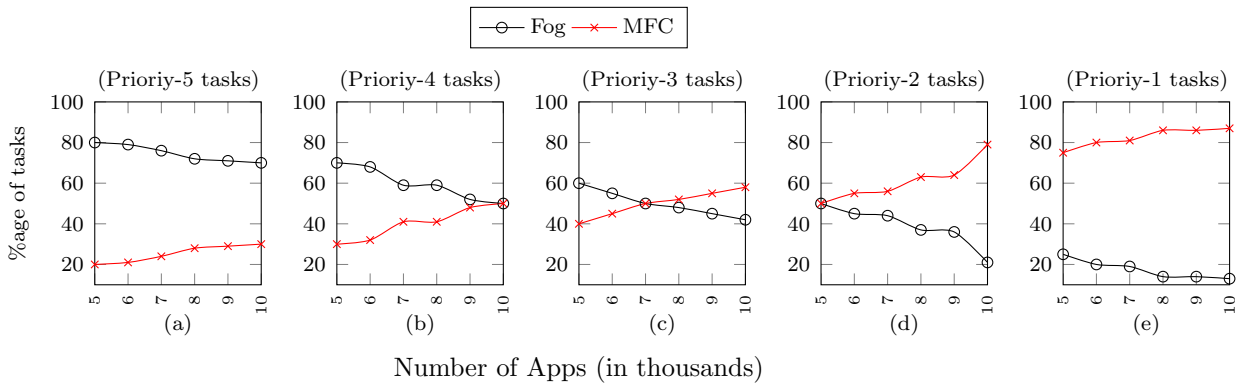


Figure 7: Percentage of tasks (of different priorities) allocated to only FNs and to MFC environments.

to 204ms when the total number of tasks increases to 100 thousand. On the contrary, the have the average latency increases from 1200ms and 1810ms when the total number of tasks with $P_i = 1$ increases from 50 thousand to 100 thousand. The similar patterns can be seen in Figure 6(b) and 6(c), where the average latency of $P_i = 5$ tasks hosted in cloud environment is approx. 61ms and 64ms when a higher weightage is given to resource demand and priority parameters, respectively. Moreover, when the $P_i = 5$ tasks are increased to 100 thousand, their average latency increases to approx. 213ms and 220ms under the same configurations, respectively. Tasks with $P_i = 1$ have the maximum latency irrespective of the weightage value of the parameters.

It is essential to investigate the percentage of tasks allocated to fog and cloud environments; the same simulation results are presented in Figure 7. The X-axis represents the number of applications ranging from 5000 to 10000. The applications consist of different priority tasks. Y-axis represents the percentage of tasks (with specific priority values) assigned to fog or cloud environments. Figure 7 consists of five sub-figures, each for specific priority values. Figure 7(a) shows the percentage of $P_i = 5$ tasks processed by the FNs that might be in the nearby location or at a multi-hop distance and by the cloud environment. Similarly, Figure 7(b) and 7(c) represent the percentage of $P_i = 4$ and $P_i = 3$ tasks, respectively, that are processed by the FNs and by the cloud environment. The simulation result in all those sub-figures shows the tasks with higher priority are given higher preference to be processed by the FN. For instance, out of all the $P_i = 5$ tasks (of 5000 applications), approximately

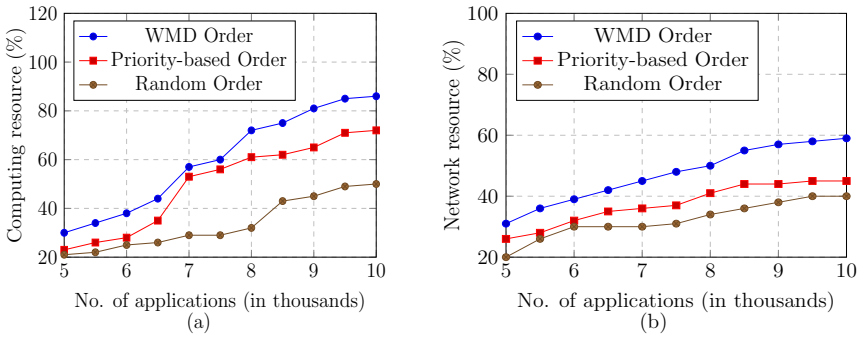


Figure 8: Impact of tasks' order on resource utilization: (a) average computing (CPU and memory) resource utilization and (b) average network resource utilization.

80% of the tasks are processed by FNs and rest 20% are processed by the cloud. However, when the number of applications increases to 10000, a decreasing percentage of the $P_i = 5$ tasks are handled by FNs, i.e., 70% of the tasks are processed by fog environment and the rest are handled by cloud, as shown in Figure 7(a).

Similarly, in the case of $P_i = 4$ tasks, it is observed that approx. 70% of the tasks are assigned to the fog environment and the rest 30% are assigned to the cloud when the total number of applications was 5000. When the total number of applications increases to 10000 (i.e. number of tasks is 100 thousand), an approximately equal percentage of $P_i = 4$ tasks are processed by FNs and by cloud environment, as shown in Figure 7(b). Figure 7(e) gives a result that is opposite to the result in 7(a). In Figure 7(e), the maximum number of tasks with $P_i = 1$ are handled by the cloud. When the number of applications was 5000, approx. 25% of the $P_i = 1$ tasks are processed by FNs and rest 75% of the $P_i = 1$ tasks are processed by the cloud. The percentage of $P_i = 1$ tasks that are sent to cloud further increases to more than 85% when the total number of applications increases to 10 thousand. This is due to the fact that, when the number of applications increases, the number of tasks increases, and their resource demand increases. However, the resource availability in the fog environment is fixed to fulfill those increasing resource demands of all the tasks. As a result, there is a higher chance of tasks being sent to the cloud environment.

Algorithm 1 decides the order of tasks to be followed while allocating the resources. This paper also investigated the effect of tasks' order on resource utilization, as shown in Figure

8. Along with the WMD (Weighted Multi-Dimensional) approach, this paper investigated resource utilization when the order of tasks is decided randomly and solely based on the priority value. Figure 8(a) and 8(b) shows the computing and network resource utilization, respectively. The X-axis represents the number of applications (ranging from 5000 to 10 thousand) and the Y-axis represents the resource utilization in percentage. In the case of random order, the computing resource utilization ranges from approximately 20% to 50% with the number of applications increasing from 5000 to 10000 (Figure 8(a)). Similarly, the network resource utilization increases from 20% to 40%, when the number of tasks increases from 5000 to 10000 (8(b)). A better result can be observed when the tasks are ordered solely based on their priority values. The computing and network resource utilization increases from 23% to 72% and 26% to 45%, when the number of tasks increases from 5000 to 10000. However, with the proposed WMD order, resource utilization can further be improved. The computing and network resource utilization increases from 30% to 86% and from 31% to 59%, respectively, which is better than the other two approaches. The user decides the tasks' priority values, which do not include tasks' resource demand and makespan values. This affects the final resource utilization, as discussed before. Similarly, in the random order of tasks, none of the parameters are given any priority/importance, resulting in a degraded resource utilization compared to the WMD-based ordering tasks strategy.

7.3. Uncertainty Analysis

The impact of fluctuating resource availability at the FNs on the resource utilization is analyzed and the results are presented in Appendix C.

7.4. Simulation time analysis

The time taken to handle different number of applications by both the algorithms is observed and the results are analyzed and presented in Appendix D.

8. Conclusions and future works

In this paper, we investigated the problem of resource allocation to users' applications consisting of multiple dependent tasks in MultiFog-Cloud environment. The proposed HeR-

AFC algorithm allows one FN to communicate with other FNs through Fog Cloud Interface. As a result, the service providers are allowed to choose a FN that is at multi-hop distance from the nearby FN to host and execute part of the application. Addressing the resource allocation problem in such environment, the aforementioned problem is first formulated as Integer Linear Programming problem and a novel heuristic HeRAFC algorithm is presented that facilitates the users to decide the importance of one task over other through priority values. For each task, the proposed algorithm explores the best hosting environment that is close to the environment where the adjacent tasks are deployed considering the makespan, latency and resource demand as the major parameters. To prove the efficiency, the simulation results of HeRAFC algorithm and other related algorithms are compared and presented. The performance of the proposed algorithm is evaluated in terms of different parameters such as average latency in fog and cloud, resource utilization in fog and cloud etc.

In a similar manner to our previous work in [15], a small experimental testbed will be used to implement the HeRAFC algorithm to make it more applicable to practical scenarios while minimizing energy consumption and improving QoS. We will consider applying the proposed HeRAFC to our proposed Indie Fog [7] architecture. Additionally, HeRAFC should incorporate both FN mobility and end user applications. The development of a simulated testbed for fog/edge computing is currently underway using an opportunistic network emulator (STEP-ONE [3]), which allows us to use different mobility models. As part of our future work, we will incorporate this dynamic behavior into HeRAFC.

Acknowledgment

This research is supported by SERB, India, through grant CRG/2021/003888. We also thank financial support to UoH-IoE by MHRD, India (F11/9/2019-U3(A)).

References

[1] C. K. Dehury, S. N. Srirama, An efficient service dispersal mechanism for fog and cloud computing using deep reinforcement learning, in: 2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID), 2020, pp. 589–598.

[2] S. Shekhar, A. D. Chhokra, A. Bhattacharjee, G. Aupy, A. Gokhale, Indices: Exploiting edge resources for performance-aware cloud-hosted services, in: 2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC), 2017, pp. 75–80. doi:10.1109/ICFEC.2017.16.

[3] J. Mass, C. Chang, S. N. Srirama, Edge process management: A case study on adaptive task scheduling in mobile iot, Internet of Things 6 (2019) 100051.

[4] Y. Liu, F. R. Yu, X. Li, H. Ji, V. C. M. Leung, Distributed Resource Allocation and Computation Offloading in Fog and Cloud Networks With Non-Orthogonal Multiple Access, IEEE Transactions on Vehicular Technology 67 (12) (2018) 12137–12151.

[5] H. Shah-Mansouri, V. W. S. Wong, Hierarchical Fog-Cloud Computing for IoT Systems: A Computation Offloading Game, IEEE Internet of Things Journal 5 (4) (2018) 3246–3257.

[6] C. Delimitrou, C. Kozyrakis, Quasar: Resource-efficient and qos-aware cluster management, SIGPLAN Not. 49 (4) (2014) 127–144. doi:10.1145/2644865.2541941.

[7] C. Chang, S. N. Srirama, R. Buyya, Indie fog: An efficient fog-computing infrastructure for the internet of things, Computer 50 (9) (2017) 92–98.

[8] H. Tran-Dang, D.-S. Kim, Frato: Fog resource based adaptive task offloading for delay-minimizing iot service provisioning, IEEE Transactions on Parallel and Distributed Systems 32 (10) (2021) 2491–2508.

[9] M. K. Mishra, N. K. Ray, A. R. Swain, G. B. Mund, B. S. P. Mishra, An adaptive model for resource selection and allocation in fog computing environment, Computers & Electrical Engineering 77 (2019) 217 – 229.

[10] L. Zhang, J. Li, Enabling robust and privacy-preserving resource allocation in fog computing, IEEE Access 6 (2018) 50384–50393.

[11] K. Li, Distributed and individualized computation offloading optimization in a fog computing environment, Journal of Parallel and Distributed Computing 159 (2022) 24–34.

[12] L. Ni, J. Zhang, C. Jiang, C. Yan, K. Yu, Resource Allocation Strategy in Fog Computing Based on Priced Timed Petri Nets, IEEE Internet of Things Journal 4 (5) (2017) 1216–1228.

[13] A. Gomez-Cárdenas, X. Masip-Bruin, E. Marin-Tordera, S. Kahvazadeh, J. Garcia, A Resource Identity Management Strategy for Combined Fog-to-Cloud Systems, in: 2018 IEEE 19th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM), 2018, pp. 01–06.

[14] V. Souza, X. Masip-Bruin, E. Marín-Tordera, S. Sànchez-López, J. Garcia, G. Ren, A. Jukan, A. J. Ferrer, Towards a proper service placement in combined fog-to-cloud (f2c) architectures, Future Generation Computer Systems 87 (2018) 1 – 15.

[15] R. Mahmud, S. N. Srirama, K. Ramamohanarao, R. Buyya, Quality of experience (qoe)-aware placement of applications in fog computing environments, Journal of Parallel and Distributed Computing (2018).

[16] L. Yin, J. Luo, H. Luo, Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing, IEEE Transactions on Industrial Informatics 14 (10) (2018) 4712–4721.

[17] Z. Liu, X. Yang, Y. Yang, K. Wang, G. Mao, DATS: Dispersive Stable Task Scheduling in Heterogeneous Fog Networks, IEEE Internet of Things Journal 6 (2) (2019) 3423–3436.

[18] Q. Li, J. Zhao, Y. Gong, Q. Zhang, Energy-efficient computation offloading and resource allocation in fog computing for Internet of Everything, China Communications 16 (3) (2019) 32–41.

[19] M. Adhikari, H. Gianey, Energy efficient offloading strategy in fog-cloud environment for iot applications, Internet of Things 6 (2019) 100053.

[20] B. Jia, H. Hu, Y. Zeng, T. Xu, Y. Yang, Double-matching resource allocation strategy in fog computing networks based on cost efficiency, Journal of Communications and Networks 20 (3) (2018) 237–246.

[21] J. Du, L. Zhao, J. Feng, X. Chu, Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee, IEEE Transactions on Communications 66 (4) (2018) 1594–1608.

[22] M. Shojafar, N. Cordeschi, E. Baccarelli, Energy-efficient adaptive resource management for real-time vehicular cloud services, IEEE Transactions on Cloud Computing 7 (1) (2019) 196–209.

[23] J. Akram, Z. Najam, A. Rafi, Efficient resource utilization in cloud-fog environment integrated with smart grids, in: 2018 International Conference on Frontiers of Information Technology (FIT), 2018, pp. 188–193.

[24] J. A. S. Aranda, R. dos Santos Costa, V. W. de Vargas, P. R. da Silva Pereira, J. L. V. Barbosa, M. P. Vianna, Context-aware edge computing and internet of things in smart grids: A systematic mapping study, Computers and Electrical Engineering 99 (2022) 107826.

[25] S. Javaid, N. Javaid, S. K. Tayyaba, N. A. Sattar, B. Ruqia, M. Zahid, Resource allocation using fog-2-cloud based environment for smart buildings, in: 2018 14th International Wireless Communications Mobile Computing Conference (IWCMC), 2018, pp. 1173–1177.

[26] I. Fatima, N. Javaid, M. Nadeem Iqbal, I. Shafi, A. Anjum, U. Ullah Memon, Integration of cloud and fog based environment for effective resource distribution in smart buildings, in: 2018 14th International Wireless Communications Mobile Computing Conference (IWCMC), 2018, pp. 60–64.

[27] D. Rosendo, A. Costan, P. Valduriez, G. Antoniu, Distributed intelligence on the edge-to-cloud continuum: A systematic literature review, Journal of Parallel and Distributed Computing 166 (2022) 71–94.

[28] T. Wang, Y. Liang, W. Jia, M. Arif, A. Liu, M. Xie, Coupling resource management based on fog computing in smart city systems, Journal of Network and Computer Applications 135 (2019) 11 – 19.

[29] S. Alamgir Hossain, M. Anisur Rahman, M. A. Hossain, Edge computing framework for enabling situation awareness in iot based smart city, Journal of Parallel and Distributed Computing 122 (2018)

226–237.

[30] A. Yasmeen, N. Javaid, O. U. Rehman, H. Iftikhar, M. F. Malik, F. J. Muhammad, Efficient resource provisioning for smart buildings utilizing fog and cloud based environment, in: 2018 14th International Wireless Communications Mobile Computing Conference (IWCMC), 2018, pp. 811–816.

Chinmaya Kumar Dehury received a Bachelor's degree from Sambalpur University, India, in June 2009 and a Master's degree from Biju Pattnaik University of Technology, India, in June 2013. He received a Ph.D. Degree in the Department of Computer Science and Information Engineering, Chang Gung University, Taiwan. He is currently a Lecturer of Distributed System, member of Mobile & Cloud Lab in the Institute of Computer Science, University of Tartu, Estonia. His research interests include scheduling, resource management and fault tolerance problems of Cloud and fog Computing, the application of artificial intelligence in cloud management, edge intelligence, Internet of Things, and data management frameworks. His research results are published by top-tier journals and transactions such as IEEE TCC, JSAC, TPDS, FGCS, etc. He is a member of IEEE and ACM India. He is also serving as a PC member of several conferences and reviewer to several journals and conferences, such as IEEE TPDS, IEEE JSAC, IEEE TCC, IEEE TNNLS, Wiley Software: Practice and Experience, etc.

Bharadwaj Veeravalli received his BSc degree in Physics, from Madurai-Kamaraj University, India, in 1987, the Master's degree in Electrical Communication Engineering from the Indian Institute of Science, Bangalore, India in 1991, and the PhD degree from the Department of Aerospace Engineering, Indian Institute of Science, Bangalore, India, in 1994. He received gold medals for his bachelor degree overall performance and for an outstanding PhD thesis (IISc, Bangalore India) in the years 1987 and 1994, respectively. He is currently with the Department of Electrical and Computer Engineering, Communications and Information Engineering (CIE) division, at The National University of Singapore, Singapore, as a tenured Associate Professor. His main stream research interests include cloud/grid/cluster computing (big data processing, analytics and resource

allocation), scheduling in parallel and distributed systems, Cybersecurity, and multimedia computing. He is one of the earliest researchers in the field of Divisible Load Theory (DLT). He is currently serving the editorial board of IEEE Transactions on Parallel and Distributed Systems as an associate editor. He is a senior member of the IEEE and the IEEE-CS.

Satish Narayana Srirama is an Associate Professor at the School of Computer and Information Sciences, University of Hyderabad, India. He is also a Visiting Professor and the honorary head of the Mobile & Cloud Lab at the Institute of Computer Science, University of Tartu, Estonia, which he led as a Research Professor until June 2020. He received his PhD in computer science from RWTH Aachen University, Germany in 2008. His current research focuses on cloud computing, mobile web services, mobile cloud, Internet of Things, fog computing, migrating scientific computing and enterprise applications to the cloud and large-scale data analytics on the cloud. He is IEEE Senior Member, an Editor of Wiley Software: Practice and Experience, a 52 year old Journal, was an Associate Editor of IEEE Transactions in Cloud Computing and a program committee member of several international conferences and workshops. Dr. Srirama has co-authored over 170 refereed scientific publications in international conferences and journals.

Appendix A. Time complexity of HeRAFC

Appendix A.1. Running time of Algorithm 1

Theorem 1. *The time complexity of the Algorithm 1 is* $\mathcal{O}(|V'|\log|V'|)$ *, where* V' *be the set of tasks.*

Proof. Algorithm 1 focuses on ordering of the tasks for deployment purpose. ST contains $|V'|$ number of tasks. Sorting of $|V'|$ tasks based on the number of out-edges depends on the sorting algorithm. This sorting time can be reduced to $\mathcal{O}(|V'|\log|V'|)$ with QUICKSORT algorithm, as in Line 1. Following this, extraction of the tasks in Line 3 would need a constant time as the list of the tasks is already in the previous step. The looping block from Line 5-8 would depend on the size of TL set. Furthermore, the sorting process in Line 10 would consume the time-complexity of $\mathcal{O}(|TL|\log|TL|)$. The extraction process in Line 13 would consume a running time of $\mathcal{O}(|TL|)$. The outer *while* loop from Line 4-16 would depend on the maximum depth of the graph and hence would iterate for $|V'|$ (which is nothing but the size of set ST) number of times. This situation would occur if all the tasks run in a sequential manner, resulting only one task in set TL in each iteration. The running time of Algorithm 1 can be calculated as $\mathcal{O}(|V'|\log|V'|) + \mathcal{O}(|V'|))$, which can be written as $\mathcal{O}(|V'|\log|V'|)$, where $|V'|$ is the number of tasks. □

Appendix A.2. Running time of Algorithm 2

Theorem 2. *The time complexity of the proposed HeRAFC algorithm (Algorithm 2) is* $\mathcal{O}(|E'|\,[\log|E'| + |E|log(|V|)]$ *, where* E *and* E' *are the set of physical edges and task edges, respectively and* V *is the number of cloud and FNs.*

Proof. Algorithm 2 depends on the output from the Algorithm 1 (Line 1), which is having a running time of $\mathcal{O}(|V'|\log|V'|)$, where $|V'|$ is the number of tasks. Calculation of availability matrix takes a constant running time. The *for* loop at Line 6 iterate for each task in the set ETL. For each task, finding the FNs (Line 6-29) that are in the communication range of user, at 1-hop distance, and at 2-hop distance would take the running time of $\mathcal{O}(|V|)$,

where $|V|$ is the set of cloud and FNs as defined in Section 5. This process will be repeated for $|V'|$ number of times, the total running time can be calculated as $\mathcal{O}(|V'| * |V|)$.

Sorting of edges of the task graph, as in Line 31, would take a running time of $\mathcal{O}(|E'| \log |E'|)$, where $|E'|$ is the number of task edges. For every task edge, an existing shortest path algorithm can be implemented. In Line 37, assuming that Dijkstra shortest path algorithm is implemented to find the shortest path from FN sh to th, the worst case running time for every task edge would be $\mathcal{O}((|V| + |E|)log(|V|))$, where $|E|$ is the number of physical edges among cloud and FNs. The looping block in Line 32 would iterate for $|E'|$ number of times and hence the total running time can be calculated as $\mathcal{O}(|E'| * (|V| + |E|)log(|V|))$. The total running time of Algorithm 2 can be calculated as follows.

$$\begin{aligned} &\mathcal{O}(|V'| \log |V'|) + \mathcal{O}(|V'| * |V|) + \mathcal{O}(|E'| \log |E'|) \\ &\qquad + \mathcal{O}(|E'| * (|V| + |E|)log(|V|)) \\ &= \mathcal{O}(|V'| \log |V'|) + \mathcal{O}(|V'| * |V|) + \mathcal{O}(|E'| \log |E'|) \\ &\qquad + \mathcal{O}(|E'| * |V|log(|V|)) + \mathcal{O}(|E'| * |E|log(|V|)) \end{aligned}$$

In case of a larger task graph, the number of task edges, $|E'|$, increases significantly as compared to the number of tasks, $|V'|$. Similarly, when the number of cloud and FNs increases, relatively the number of physical edges, $|E|$, increases significantly. Considering this scenario, the total running time can be calculated as follows.

$$\begin{aligned} &\mathcal{O}(|E'| \log |E'|) + \mathcal{O}(|E'| * |E|log(|V|)) \\ &\qquad = \mathcal{O}(|E'| \left[\log |E'| + |E|log(|V|)\right]) \end{aligned}$$

□

Appendix B. Simulated MFC environment and Applications

This section provides an in-depth understanding of programmatically generated application and MFC environment that following several distribution functions. The entire simulation environment basically consists of two primary components: Applications and the

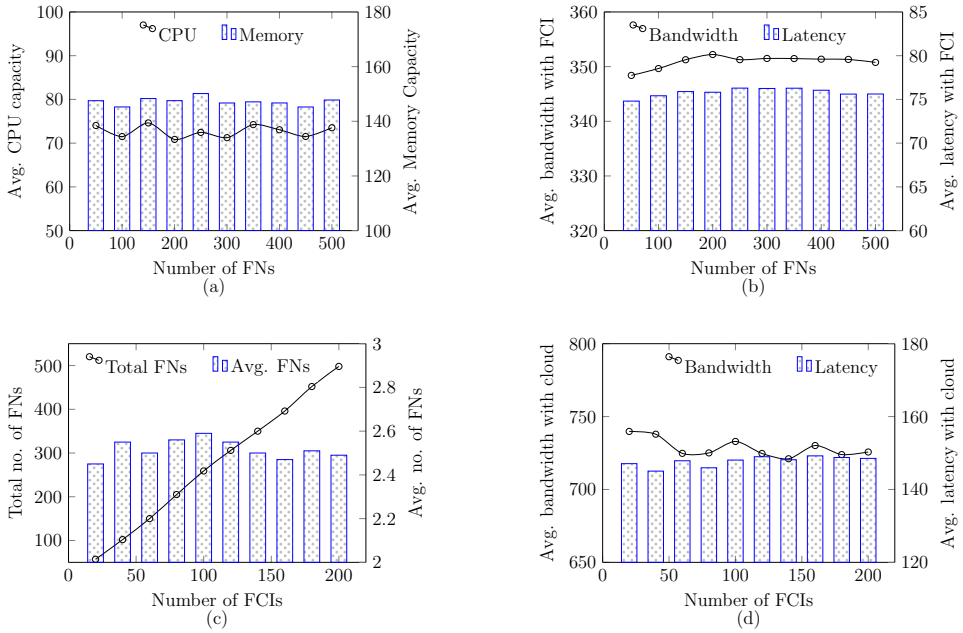


Figure B.9: Analysis of the MFC environment (i.e. FNs, FCIs, and cloud) and their resource availability and connectivity.

fog-Cloud environment. The resource configuration of the FCIs, fog, cloud and the users' applications are presented in JSON file. Separate JSON files are created for each user's app, including the tasks list, edges list, resource demand of tasks and edges, and other related information.

Appendix B.1. Fog-Cloud Environment

The MFC environment consists of 500 FNs that are connected to 200 FCIs. Each FCI is connected to at least one FN, whereas a single FN is connected to precisely one FCI. For each FN, the total amount of CPU capacity ranges from 50 through 100 and the values are assigned by following a random distribution. The total memory (RAM) for each FN is assigned randomly ranging from $200GB$ to $400GB$, where the unit GB represents GigaByte. The approximate computation power of a FN is calculated as MIPS (Million instructions per second), which ranges from 3000 to 5000. As discussed before, each FN is connected to exactly one FCI. The maximum network bandwidth available between a FN and the corresponding FCI ranges from $300Mbps$ through $400Mbps$. In this paper, it is considered that a

FCI may directly communicate with another FCI. In such a situation, there exist a network bandwidth capacity among FCIs, which ranges from $400Mbps$ through $1000Mbps$. Similarly, the bandwidth between FCIs and the cloud range from $400Mbps$ through $1000Mbps$. The network bandwidth unit is in Mbps (Mega bits Per Second). It is assumed that latency value between any FN and the corresponding FCI is small than the latency among FCIs and between FCIs and cloud. Latency values are assigned to the physical links by following a uniform distribution. The latency (in millisecond) between FN and FCIs ranges from $50ms$ through $100ms$. Similarly, the latency values for rest of the physical links range from $100ms$ through $200ms$. Technically, a task can be hosted on 1-hop or at a multi-hop distance. However, in the current implementation maximum number of hops is set to 2. This means there are a maximum of two FCIs and no cloud are present in the physical path between any two FNs that are used to host user's tasks.

Figure B.9(a) and B.9(b) present the computing resource and network resource distribution, respectively, among the FNs. Figure B.9(a) shows the average CPU and memory resource availability among the FNs. Though the minimum and maximum CPU resource assigned to a FN is 50 and 100, respectively, the average CPU resource available is approximately 75 and 73 when there are 50 and 500 FNs, respectively. Similarly, the average memory resource available ranges between $145GB$ and $150GB$ with the same number of FNs, as shown in Figure B.9(a). Figure B.9(b) represents the relationship between the number of FNs and the average bandwidth and latency with the connected nearby FCI. The x-axis of the figure represents the number of FNs, while the y-axis represents the average bandwidth and average latency with FCI as two separate series. The figure shows that as the number of FNs increases, the average bandwidth with FCI remains relatively constant, fluctuating around the value between $348Mbps$ and $353Mbps$. However, the average latency with FCI also shows a slightly increasing trend, from $75.4ms$ for 50 FNs to $75.6ms$ for 500 FNs, with a maximum of 76.2 for 350 FNs. This means that as the number of FNs increases, the average latency with FCI slightly increases, but it remains relatively low overall. Figure B.9(b) indicates that increasing the number of FNs has little impact on the average bandwidth with FCI, while it slightly increases the average latency with FCI.

Figure B.9(c) and B.9(d) show the relationship of the FCIs with the number of FNs and the network resource availability with the cloud. In Figure B.9(c), the x-axis denotes the number of FCIs, while the left y-axis represents the total number of FNs and the right-side y-axis represents the average number of FNs that are connected to the FCI. This shows how the FCIs are distributed and connected with all the 500 FNs in the simulated environment. The total number of FNs connected to the FCIs is constantly increasing and as expected, 498 number of FNs are connected to at least one FCI. In the simulated environment, only two FNs are directly connected with cloud. In other words, two FNs have no nearby FCI. The average number of FNs that are connected to the FCIs ranges between 2.4 and 2.6, as shown in Figure B.9(c). Each connection between an FCI and the cloud is associated with a certain amount of network bandwidth and latency, as shown in Figure B.9(d). The minimum and maximum network bandwidth available with any pair of FCI and cloud connection is $400Mbps$ and $1000Mbps$, respectively. However, the average bandwidth available among FCIs and the cloud ranges between $720Mbps$ and $740Mbps$. Similarly, in the simulated environment, the average latency between FCI and cloud ranges between $145ms$ and $150ms$, as shown in Figure B.9(d).

Appendix B.2. Users Application

The maximum number of applications is set to 10000. For each application, random number of tasks are generated ranging from 4 through 12. However, the total maximum number of tasks for the entire simulation is set to $100,000$. For each task, the CPU and memory demand range from $1-4CPUs$ and $100MB-1000MB$, respectively. The average makespan of each task is $350ms$ with minimum and maximum makespan of $10ms$ and $1000ms$, respectively. The network bandwidth demand of edges is ranging from $100Mbps$ and $200Mbps$. Similarly, the average latency demand among tasks ranges between $10ms$ and $50ms$. The unit of latency is millisecond (ms). The minimum and maximum priority values of a task are 1 and 5, respectively. The bandwidth and latency demand of a task ranges between $100-200Mbps$ and $10-300ms$, respectively. The number of edges within an application is decided by a link probability of 0.6. This indicates the connectivity probability

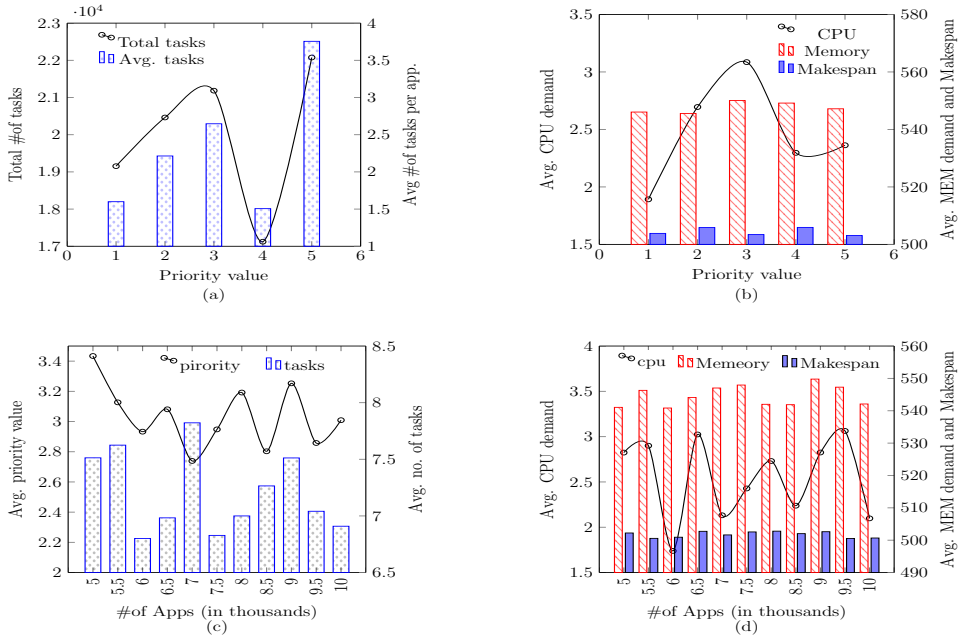


Figure B.10: Analysis of the applications and their tasks including the priorities and resource demand.

among two task.

Figure B.10 gives an insight into the applications and their tasks, including the priorities and the resource demand. Figure B.11a shows the total and average number of tasks with different priority values. The priority values are assigned to the tasks in a programmatic manner. From Figure B.10(a), it can be observed that there are a total of 19154, 20463, and 21184 number of tasks having priority 1, 2, and 3. With a priority value 4 there are a total of 17123 number of tasks. In the entire simulation, the highest number of tasks have a priority value 5, i.e. 22076 tasks. The average number of tasks per application with a specific priority value is also analyzed and presented in the same figure. An average of 1.5 number of tasks per application are with priority 4, which is the lowest. On the other hand, an average of 3.75 number of tasks per application have the priority 5, which is the maximum, as shown in Figure B.10(a). Figure B.10(b) gives the tasks' resource configuration and makespan information with their priority. It can be seen that the tasks with priority 3 are having highest CPU and memory demand, which is 3 and $550MB$, respectively. Moreover, such tasks have comparatively lower makespan requirement, which is $503ms$. The tasks with priority 2 and

4 have the higher makespan requirement of 505.8ms and 505.9ms, respectively.

The applications with average priority and the average number of tasks are also analyzed in Figure B.10(c). It can be observed that the average priority of the applications ranges between 2.5 and 3.5. The average number of tasks of 5000 applications is 7.5, whereas this number is reduced to 6.9 when all the 10 thousand applications are taken into account. The applications are further analyzed in terms of their resource (CPU, memory) demand and makespan requirement in Figure B.10(d). The average CPU demand with a different number of applications ranges between 1.7 and 3.0, whereas the memory demand ranges between 540MB and 550MB. The average makespan requirement of all 10 thousand applications is 500.6ms. The pattern shows that the distribution of CPU demand, memory demand, and makespan requirements are independent of each other.

Appendix C. Uncertainty Analysis

Appendix C.1. Uncertain resource availability impact on utilization

Figure C.11 represents the relationship of fluctuation of resource availability, including network bandwidth and latency among FNs, FCIs, and cloud with their utilization. In other simulation instances, the HeRAFC algorithm takes the resource availability values of the entire environment, which remains unchanged until the algorithm handles the last application. For example, suppose the CPU availability of a FN is 75% of its maximum capacity before HeRAFC starts processing the first application. In that case, CPU availability remain unchanged until the last application, provided no task is assigned to that FN.

It is implemented in a way that when the proposed algorithm starts, the resource availability at FNs and at the cloud is not 100% to represent a real-life scenario. In this simulation, the distribution of resource availability follows a random distribution. As shown in Figure D.12c, the total simulation time required to assign 10000 applications is approximately 230sec. However, in this simulation, we randomly modify the resource availability of FNs and cloud at an interval of 20sec, as shown in Figure C.11. The X-axis of all subfigures represents the time interval at which the resource availability of the entire environment is modified.

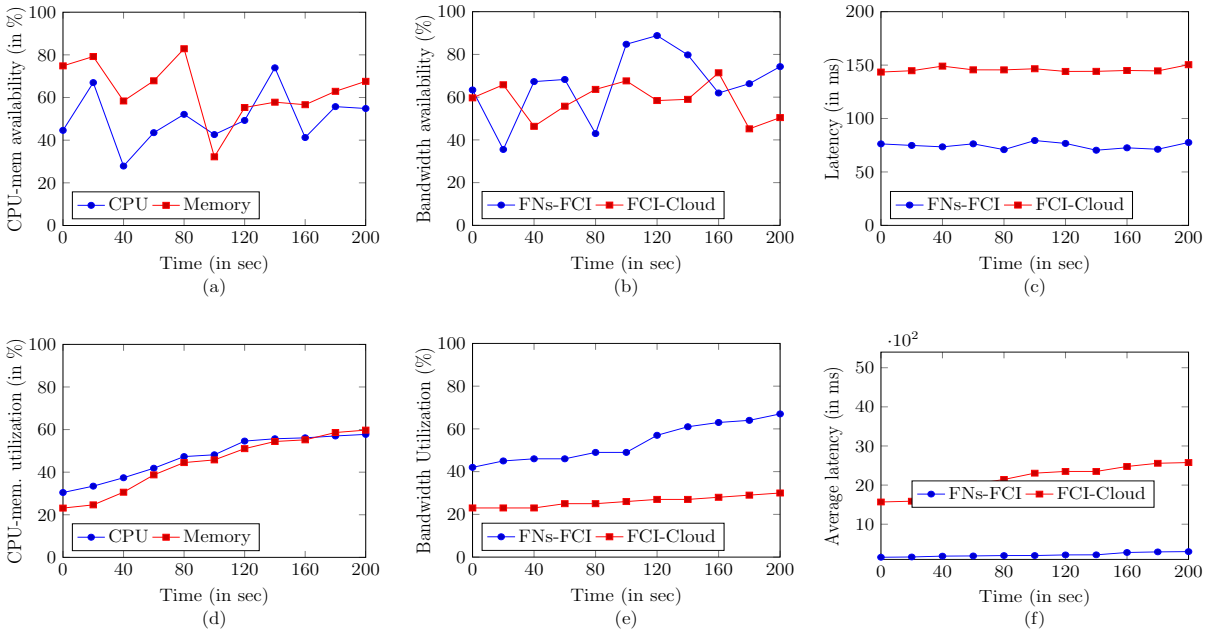


Figure C.11: Comparison of only CLOUD resource utilization (a) CPU and memory availability, (b) Bandwidth availability, (c) Latency (in ms), (d) CPU-memory utilization (in %), (e) Bandwidth Utilization (%), (f) Average latency of tasks (in ms).

Figure C.11(a) shows the modified CPU and memory resource availability at different time intervals. It can be seen that the initial CPU and memory resource availability at FNs was 45% and 74%. After 20sec, the CPU and memory configurations of the FNs were changed to 67% and 79%, respectively. A similar pattern can be seen in Figure C.11(b), where bandwidth among FNs and FCIs (Blue solid line) and bandwidth among FCI and cloud (Red solid line) resource availability is modified. Bandwidth availability among FNs and FCI ranges between 35% and 89% across the entire simulation. On the other hand, bandwidth among FCIs and cloud ranges between 45% and 70%. As shown in Figure C.11(c), the latency value, calculated in milliseconds, among FNs and FCI is modified at every 20sec, and the value varies between 70ms and 80ms. Similarly, the latency among FCIs and cloud, at every 20sec, ranges between 140ms and 150ms.

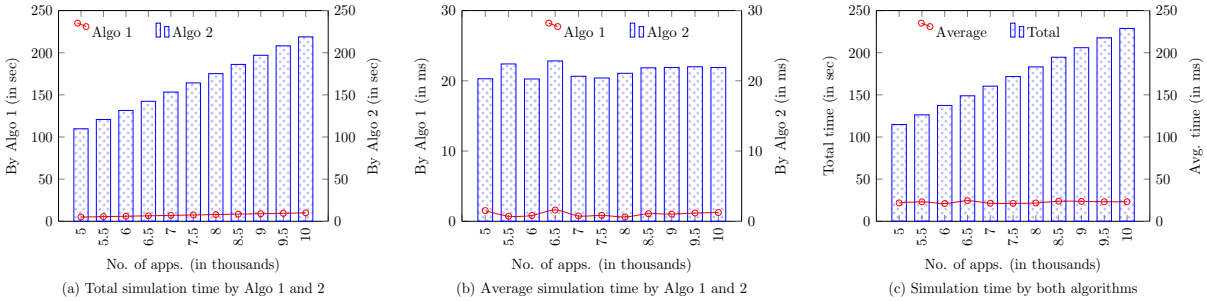
Under the above uncertain resource availability, Figures C.11(d) - C.11(f) represent the impact on resource utilization. It is observed that such uncertain conditions and fluctuated resource availability have a higher negative impact on utilization. Figure C.11(d) shows the

CPU and memory utilization of the FNs. In comparison with the results in Figure 3a and 3b, it is observed that the CPU and memory resources are being underutilized by approximately 10%. This can be observed in the figure where the maximum CPU and memory utilization are less than 60% each. At the same time, this utilization is observed to be more than 65%, as in Figure 3a and 3b.

A similar pattern is observed in the case of network bandwidth utilization. The maximum bandwidth utilization among FNs and among FCI and cloud is observed as 78% and 40%, in Figure 3c and 4c, respectively. However, this utilization is reduced by more than 10% when the bandwidth resource availability fluctuates, as shown in Figure C.11(e). The bandwidth utilization is less than 67% among FNs and FCI, and the utilization among FCI and cloud is less than 30%. The uncertain latency among FNs, FCI, and cloud is also analyzed and presented the results in Figure C.11(f). The latency of the network communications among FNs and FCIs increases from $155ms$ to $300ms$ while processing all the applications by HeR-AFC algorithm, which is more than 30% increase from the previous result, shown in Figure 5. Similarly, the minimum latency among FCIs and cloud is 1570%, which is more than approximately 30% of the previous result, as shown in Figure 6. Overall this observation indicates that latency is the worst-performing parameter among others when the network communication becomes very unstable and fluctuates over time.

Appendix D. Simulation time analysis

Figure D.12 helps in visualizing the performance of Algorithm 1 and 2 in terms of simulation time taken for different numbers of applications, ranging from 5000 to 10000. The number of tasks ranges from 4 to 12 per application. However, the maximum number of tasks is set to 100 thousand. The x-axis represents the number of applications, while the Y-axis in all the sub-figures shows the simulation time. The results suggest that the total simulation time for both algorithms increases as the number of applications increases. Figure D.12(a) represents the time taken by Algorithm 1 and 2 to process a certain number of applications. For instance, as in Figure D.12a, when there were 5000 applications, Algorithm 1 takes a total of approximately 5 seconds, while Algorithm 2 takes more than 100 seconds. Similarly,



(a) Total simulation time by Algo 1 and 2

(b) Average simulation time by Algo 1 and 2

(c) Simulation time by both algorithms

Figure D.12: Simulation time by Algorithm 1, Algorithm 2 and by both combined.

when the number of applications increases to 10000, Algorithm 1 takes approx. 10 seconds, while Algorithm 1 takes more than 220 seconds. It is observed (also expected) the total simulation time taken by Algorithm 2 is consistently higher than that of Algorithm 1, across all tested scenarios.

Similar to the above results, the average time to handle an application by Algorithm 1 and 2 is also observed and presented in Figure D.12(b). Algorithm 1 takes an average of $1.5ms$ when the number of applications is 5000, whereas Algorithm 2 takes an average of more than $20ms$ for the same number of applications, as shown in Figure D.12(b). Algorithm 1 and 2 took an average of $1.25ms$ and $21.9ms$ per application, respectively, when the number of applications increases to 10000. Unlike Figure D.12(a), presenting the time taken by individuals algorithms, Figure D.12(c) presents the average and total time taken by both the Algorithms 1 and 2. Figure D.12(c) shows that approximately $114sec$ is required to process 5000 applications, with an average of $21ms$ per application. When the number of applications increases to 100000, a total of $228sec$ (with an average of $23ms$ per application) is required to assign those applications to fog and cloud environments, as shown in Figure D.12(c). Overall, the Figure D.12 provides a comparison of the simulation time performance of two algorithms for different application scenarios.