


Cost-Effective Scheduling in Fog Computing: An Environment Based on Modified PROMETHEE Technique


Shefali Varshney

(Jaypee University of Information Technology, Solan, India

 <https://orcid.org/0000-0003-0099-8192>, shefali1926@gmail.com)


Rajinder Sandhu

(Jaypee University of Information Technology, Solan, India

 <https://orcid.org/0000-0001-8658-6089>, rajsandhu1989@gmail.com)

P. K. Gupta

(Jaypee University of Information Technology, Solan, India

 <https://orcid.org/0000-0003-0416-7097>, pkgupta@ieee.org)

Abstract: With the rising use of Internet of Things (IoT)-enabled devices, there is a significant increase in the use of smart applications that provide their response in real time. This rising demand imposes many issues such as scheduling, cost, overloading of servers, etc. To overcome these, a cost-effective scheduling technique has been proposed for the allocation of smart applications. The aim of this paper is to provide better profit by the Fog environment and minimize the cost of smart applications from the user end. The proposed framework has been evaluated with the help of a test bed containing four analysis phases and is compared on the basis of five metrics- average allocation time, average profit by the Fog environment, average cost of smart applications, resource utilization and number of applications run within given latency. The proposed framework performs better under all the provided metrics.

Keywords: QoE, Fog environment, Cost-effective scheduling, Smart applications, MCDM

Categories: C.4, C.5, F.2.1, L.5.0, L.7.0

DOI: 10.3897/jucs.90429

1 Introduction

The extension of internet has introduced the world to the development of Internet of Things (IoT). This environment comprises of several devices such as home appliances and sensors that can interact with one another. The working of these devices includes collecting information and generating data which should be processed for decision making [Singh et al., 2020]. This data generated from the IoT devices can be processed for service distribution to its users with the help of cloud computing. A cloud computing platform is helpful in providing storage resources and computing facilities and has engineered the goal of being economical, robust, readily available and flexible. The data thus generated requires the control to transmit data and receive instructions. Whereas, sending a huge amount of data to cloud servers will overload the network resulting in data transfer overhead and latency issues [Martinez et al., 2020]. This

situation results in a hostile environment that relies on cloud servers for the real time applications such as self-driving cars and virtual reality games. This issue in cloud computing was solved with the arrival of a new platform known as Fog computing—a computing environment is well renowned for its heterogeneity and ability to provide services closer to the edge of the network as shown in Figure 1. Figure 1 represents the 3-layer architecture of Fog computing environment. IoT smart application layer consists of smart car, smart light, smart watch, etc. These devices generate data which is sent to the cloud for processing, but due to delay in the response of the service providers a Fog computing layer was introduced. Therefore, the data of IoT smart application layer is sent to the Fog computing environment layer for intermediary storage and processing. In this layer all the functions are performed by the Fog nodes reducing latency and improving performance. The processed data is then forwarded back to the smart devices for better management of their data. The Fog node send the data to the cloud server for further processing. The data is then stored in a cloud centre database. The number (1) on arrow shows the IoT smart applications requesting data from the cloud via Fog computing environment. The number (2) on arrow depicts the data of smart application not requiring immediate response is sent for processing to the cloud computing environment layer. The (3) on arrow shows the processed data is forwarded to the IoT smart applications. The (4) on the arrow depicts the smart application processed data that is sent back to the requiring smart applications. Thus, due to latency the computing resources relocate from centralized clouds to distributed Fog devices once placed near the data sources.

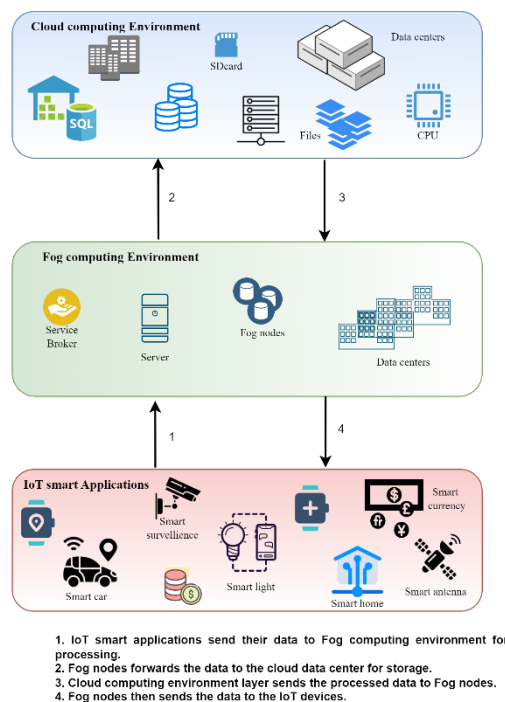


Figure 1: Fog Computing Environment Architecture

With the large number of incoming tasks at the Fog computing environment, the smart applications are assigned random resources [Goudarzi et al., 2022]. Random allocation of resources sometimes results in their underutilization and smart applications not performing well. As a consequence, the smart applications are not able to provide better QoE services to its users. Thus, the Fog resources need to be organized in a manner resulting in better performance of smart applications [Sidhu and Singh, 2019]. This problem needs to be solved by techniques wherein Fog environments are organized based on priority. To resolve this issue, we have used the MCDM technique since it handles complex problems like these and are famous for providing a list of ranking of alternatives to select the best optimal alternative. Several multi criteria decision analysis (MCDA) / multi criteria decision making (MCDM) methods have been presented in recent years to aid in the selection of the best compromise alternatives. In this paper, modified (Preference Ranking Organization Method for Enrichment Evaluations) PROMETHEE-II technique has been used for the allocation of Fog environment to the smart applications. The main objectives of this paper are shown below:

- PROMETHEE-based scheduling approach is discussed in this paper for better allocation of smart application.
- Proposed cost-effectiveness-based scheduling algorithm for Fog computing environment.
- Proposed Framework is divided into three stages for better allocation to the smart applications.

Further, this paper is categorized into various sub-sections where Section 1 focuses on fog computing environment and MCDM based techniques. Section 2 provides the insights about the previous studies carried out in the field of scheduling and frameworks concerning to cost effective Fog computing. Section 3 focuses on materials and methods, proposed framework, datasets, and proposed modified PROMETHEE -II algorithm. Section 4 provides the detail of experimental setup and perform latency mapping analysis and cost mapping analysis. Section 5 includes the various results about the performance analysis of the proposed approach. Finally, section 6 concludes this work.

2 Related Work

Cost maintenance and pricing policies of the internet providers have already been a point of concern in the field of cloud computing [Mahmud et al., 2020]. But Fog computing varies from cloud computing in terms of its several resource-constrained and heterogeneous Fog nodes. Therefore, in this paper an effective literature survey has been conducted which is divided into two subsections. First sub section discusses about the literature related to cost-effective scheduling in Fog computing environment whereas the second sub-section addresses the MCDM techniques considered for scheduling in various computing environment.

2.1 Cost effective scheduling in Fog computing environment

Pham et al. [Pham et al., 2017] have designed a scheduling algorithm known as Cost-Makespan which is an aware scheduling heuristic that balances the cost and application execution of cloud resources for the user. The authors also proposed a reassignment plan on the basis of direct a cyclic graph. Experimental results show that the proposed approach is more efficient compared to others. Yao et al. [Yao et al., 2017] have presented a heuristic algorithm and an integer linear programming form to address the issue of server placement in a cost-effective manner. The simulation studies justify the efficiency of the proposed algorithm as it provides results much near to the ideal solution. Mann [Mann, 2022] has examined the effects of coordination and decentralization on the optimization results of four individual approaches. Experimental results demonstrate the best results for several problem instances as well as decentralization combined with coordination. Fan et al. [Fan et al., 2017] proposed a deadline aware scheduling mechanism in a tiered IoT framework for Fog computing. The authors formulated the task scheduling problem as a knapsack problem and further proposed a solution based on ant colony optimization heuristic for the algorithm. Extensive results show that the system performance has improved compared to available heuristics. Yu-Jiang and Zou [Yu et al., 2017] have designed a parallel and distributed load balancing algorithm to reduce the cloud data-center's operational cost through Fog devices. The presented algorithm is based on proximal Jacobian altering direction method. The simulation result shows the efficiency of the proposed algorithm. Liu et al. [Liu et al., 2018] have worked on the problem of searching for a server configuration in order to meet its resource needs while minimizing its cost. The authors worked on this NP hard problem and designed a partial rounding algorithm (PRA). Chern-off-bound was also proposed by the authors and applied to the PRA. The experimental results show that the proposed algorithm finds solutions close to the ideal one.

2.2 MCDM Techniques

Mishra et al. [Mishra et al., 2019] have proposed an adaptive MCDM model to acquire the best solution in scalable and dynamic environment. The proposed approach takes only $O(m)$ time to assign ranks to the alternatives. The performance of the proposed MCDM model is found to be better than other MCDM methods. Sidhu and Singh [Sidhu and Singh, 2019] have designed a PROMETHEE based selection technique in order to select trustworthy Cloud data base server. The evaluation of the technique has been presented in this paper using a case study consisting of real cloud data from Cloud Harmony reports. The author has used this report in form of dataset for trust evaluation and cloud database server selection. The experimental results show the technique performs well in terms of real cloud environment. Hosseini et al. [Hosseini et al., 2022] presented a scheduling algorithm namely PQFAHP in Fog computing environment. The authors have used parameters such as completion time, deadline criteria, energy consumption, and RAM. In this paper proposed technique is used for prioritizing multi criteria and combining several other priorities. The experimental result displays that the proposed approach performs better in terms of service level and considered benchmarks. Hussain and Merigo [Hussain and Merigó, 2022] addressed the issue of cloud service selection with the help of CQoES repository framework. This method

uses PROMETHEE-II technique in which the alternatives are evaluated based on the consumers custom weighted QoS attributes. In 2017 Ni et al. [Ni et al., 2017] proposed a resource allocation strategy for users, such that they can choose their resources from the batch of pre-allotted resources. The authors' proposed strategy examines the time cost and price cost to complete a task of both fog resources respectively. The experimental strategies demonstrate that the proposed algorithm can obtain high efficiency in provision with the price and task completion time. Afrin et al. [Afrin et al., 2017] developed a non-linear programming solution for the profit aware resource allocation problem and QoE. Two scheduling algorithms—First fit satisfaction and profit algorithm—were respectively designed. The simulation is on toolkit and demonstrates that the proposed system outperforms the average waiting time, service provider profit and user QoE. Kiani and Ansari [Kiani and Ansari, 2017] designed a hierarchical model based on LTE advanced back-haul network. The proposed model is designed in three hierarchical levels and introduces the concept of shallow, field, and deep cloud-lets. The authors also allocate the communication resources to satisfy the users' QoS.

3 Dataset Formation

This experiment study uses a MCDM approach for cost-effective scheduling in the Fog computing environment. The dataset used is synthetically generated for the evaluation of the proposed framework. The results are explained by comparing with the other two Fog models with the help of five metrics. This experiment is run on Python framework 3.7. The further evaluation of the proposed framework is explained in detail in the following subsections.

3.1 Synthetic Data generator

For the evaluation of the proposed framework, a Fog computing dataset is required which consists of attributes such as RAM and storage requirements, no. of CPU cores required, and up-link and down-link latency, among others. A detailed search has been constructed in search of dataset on sites such as data world, Kaggle, etc. But we were not successful in finding a dataset meeting our requirements. Therefore, a synthetic Fog environment dataset is created with the help of random processes for Fog environment and smart application QoE attributes. Thus, in this paper, we have a designed a Fog environment dataset with parameters such as up-link bandwidth, down-link bandwidth, number of cores, RAM and storage requirements, up-link latency, down-link latency, time to finish, and cost, which is presented in Table2. The dataset is generated with the help of simulation parameters which are listed in Table1. The proposed framework is implemented in Python framework 3.7 and the input to this script is provided in the form of an Excel file.

Simulations Parameters	Fog Environment	Smart Application
Uplink Bandwidth (gbps)	1 – 100	1 – 100
Downlink Bandwidth (gbps)	1 – 100	1 – 100
RAM requirement (gb)	2 – 10	2 – 10

Storage requirement (gb)	2-16	2-16
Uplink Latency (ms)	50 – 100	50 – 100
Downlink Latency (ms)	50 – 100	50 – 100
Number of Cores required (CPU cores)	10 – 80	10 – 80
Time to finish (ms)	60 – 100	60 – 100
Cost(\$)	1 – 20	1 – 10

Table 1: Simulation Parameters considered for Data generation

Fog Environment(F.E)	RAM requirement (gb)	Storage requirement (gb)	No. of cores (CPU cores)	uplink latency (micro sec)	downlink latency (micro sec)	uplink bandwidth (gbps)	downlink bandwidth (gbps)	Time to Finish (micro sec)	Cost (\$)
F.E1	7	8	50	555	2677	27	74	81	3
F.E 2	8	10	43	335	2426	85	56	75	19
F.E 3	7	10	61	717	3504	97	78	11	28
F.E 4	6	5	17	501	3306	56	59	43	10
F.E 5	7	6	59	451	3123	29	35	76	16
F.E 6	7	2	43	688	4020	77	40	30	30
F.E 7	2	10	41	741	3297	25	74	43	12
F.E 8	2	9	68	506	3003	39	98	95	35
F.E 9	5	7	24	392	3439	75	36	27	25
F.E 10	5	10	21	659	3622	22	81	20	21
F.E 11	4	2	46	438	2896	86	69	35	16
F.E 12	6	10	45	400	2779	100	22	85	21
F.E 13	3	6	19	526	4048	76	6	40	17
F.E 14	6	2	63	643	2667	15	32	43	8
F.E 15	3	2	57	468	2807	9	38	80	34
F.E 16	8	3	28	496	2459	55	6	63	17
F.E 17	5	2	61	366	3636	72	79	84	29
F.E 18	6	8	24	624	3450	13	17	31	4
F.E 19	3	3	57	512	2783	66	58	83	14
F.E 20	6	2	43	305	3666	44	35	67	3

Table 2: Fog Environment Dataset

Smart Application (S.A)	Data To Transfer (MB)	Avg. Runtime of Task (sec)	Total time in which to complete (sec)	Total Task (no. of task)	Total Core require (CPU cores)	RAM (gb)	Storage (gb)
S.A 1	40	9	265	46	7	4	5
S.A 2	25	5	533	15	5	6	6
S.A 3	90	7	237	34	11	2	5
S.A 4	99	6	55	6	5	5	9
S.A 5	50	8	74	34	5	7	8
S.A 6	30	6	393	51	4	2	7
S.A 7	32	10	362	68	11	3	9
S.A 8	67	7	181	47	1	6	3
S.A 9	93	8	35	10	3	3	6
S.A 10	53	8	100	10	3	8	3
S.A 11	43	8	448	42	1	3	10
S.A 12	78	5	376	57	6	7	5
S.A 13	96	6	93	49	5	6	3
S.A 14	29	5	289	61	4	7	10
S.A 15	53	7	304	51	6	8	4
S.A 16	40	7	310	35	10	4	2
S.A 17	32	7	565	63	7	7	6
S.A 18	91	10	383	22	1	4	3
S.A 19	72	10	375	65	6	4	8
S.A 20	82	6	76	41	8	6	8

Table 3: Smart Application Dataset

For evaluating the proposed framework which is divided in three stages, smart application parameter values are required. Therefore, we have generated a smart application dataset with QoE parameters like data to transfer, avg. runtime of task, total time in which to complete, total task, total core required, RAM requirement, and storage requirement, which is listed in Table 3. This dataset values helps in better allocation of Fog environment to the smart applications meeting their requirements.

3.2 Proposed Framework

This section provides a brief overview of the proposed framework for the allocation of Fog environment resources meeting the needs of latency sensitive applications. The QoE parameter considered for the allocation of Fog environment are RAM and storage requirements, number of cores, up-link latency, down-link latency, up-link bandwidth, and down-link bandwidth. Whereas, for the smart applications, these are minimum network bandwidth, data to transfer, average run-time of task, total time in which to complete the task, total core requirement, RAM, and storage for smart applications. The

proposed algorithm is divided into three stages that are followed in the sequence for allocation of the Fog environment to any smart application. The three stages of the algorithm are: Resource mapping stage, Latency mapping stage, and Cost mapping stage. The workflow of the proposed framework is shown in Figure 2.

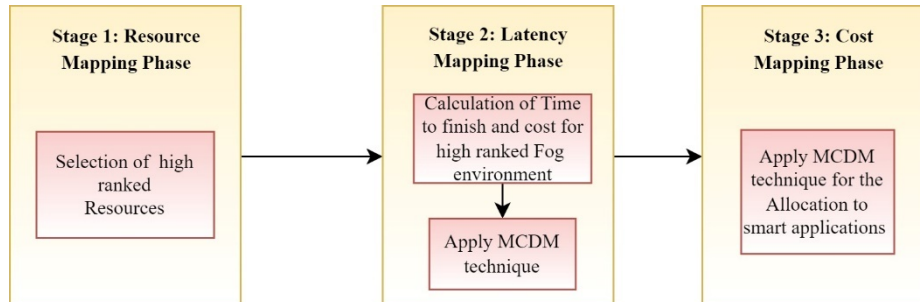


Figure 2: Component diagram of Proposed Framework

3.2.1 Resource Mapping Stage

In the first stage, Fog environments in the resource pool for filtering the resources are considered on the basis of smart application requirements. The attributes considered for filtering the Fog resources are RAM requirement, Storage requirement, and number of CPU cores. All the attributes are assessed for better allocation of resources, such as RAM and storage, which tells the size required for storing the data, total core requirement for smart applications gives the information about the CPU cores required. In this stage, Algorithm 1 is used for filtering out the Fog resources and the steps are as follows:

Initially for all the values of $F.E[]$, it checks whether all the resources present in the $F.E[]$ satisfy the requirement of the $S.A[]$. The requirements are checked on the QoE parameters like RAM requirement, storage requirement, and the number of core requirements. After this step, the $F.E[]$ matching the needs of $S.A[]$ are stored in $Resr_Mapp[]$ buffer for further evaluation. The remaining Fog environment values are discarded which do not fulfill the requirement of that particular smart application. This algorithm will run for every smart application demanding the Fog resources for its services.

Algorithm 1: Resource Mapping Algorithm

Input: $F.E[]$, $SA[]$

Output: $Resr_Mapp[]$

Begin

for values of $F.E[]$

if resources of $FE[]$ satisfy the requirement of $SA[]$

 Store the $FE[]$ values in $Resr_Mapp[]$

else

 Discard the $FE[]$ values which are not satisfying the requirements of $SA[]$

end if

end for

End

3.2.2 Latency Mapping Stage

In the second stage, the selected Fog environments from Algorithm 1 are provided as an input to the Algorithm 2. For all the values of the Fog environment that are stored in Resr_Mapp[], Algorithm 2 is applied for the further selection of Fog environment which will help in allocation of Fog environment. For the allocation process only scheduling time T_s , data transfer time T_d by Eq. 1, and the boot time T_b is evaluated. Here, scheduling time is referring to as time required in assigning the process to the applications. Whereas, the boot time is considered as the time which server takes to process another request by the application. Boot time would be less if the server is already running and if the server is not processing any application, then it will take more boot time to process another application request. Here, time to finish refers to the time taken to complete the allocation process. The time to finish (λ) for Resr_Mapp[] is calculated with the help of Eq. 2.

$$T_d = (\text{Time in which data is sent} * \text{upward bandwidth}) + (\text{Time in which data is received} * \text{downward bandwidth}) \quad (1)$$

$$\lambda = (\eta_r * (\eta / m)) + T_s + T_b + T_d \quad (2)$$

The working of Algorithm 2 is explained with the help of four steps: Firstly, the input is provided in Res_Mapp[] buffer and time to finish is calculated with the help of Eq. 2. After calculating this, all the values are stored in ToF[]. In the next step, Algorithm 3 is applied for all the values in ToF[] and store the rank values in RFE[]. Algorithm 3 is explained in sub-section 3.2.3 below. RFE[] buffer now contains all the Fog environment in their rank order. In this paper for better allocation, only a limited percentage of the Fog environment will be considered further for the allocation process. This process saves a lot of time when compared with the one that allocates all the smart applications. In order to select the Fog environment a small percentage needs to be selected that is n. For this algorithm, we had considered top 10 low rank values for the selection of Fog environment. After this the final selected Fog environment are stored in Latency_Mapp[] buffer for further processing.

Algorithm 2: Latency Mapping Algorithm

Input: Resr_Mapp, ToF[]

Output: Latency_Mapp[]

Begin

for all values in Resr_Mapp[]

Calculate $\lambda = (\eta_r * (\eta / m)) + T_s + T_b + D_t$

Store λ in ToF[]

for all values in ToF[]

Apply Algorithm 3 on ToF[]

Store the result values in RFE[]

for all values in RFE[]

Select the top n low rank values in RFE[]

Store the selected top low rank values in Latency_Mapp[]

Discard the rest of the RFE[] values

end for
 end for
 end for
 End

3.2.3 Modified PROMETHEE-II Technique

The PROMETHEE technique was first developed by BransVincke and Mareschal [Brans et al., 1986] and belongs to a type of outranking methods. The PROMETHEE technique is further of two types which are PROMETHEE-I and PROMETHEE-II. The PROMETHEE-I [Khansoltani et al., 2022] technique provides partial ranking to the alternatives therefore it is not considered for the evaluation purpose. Whereas, the PROMETHEE-II technique provides the total ranking to the alternatives and is considered more for evaluation. In this paper, we have proposed a modified version of PROMETHEE-II technique for the evaluation of the proposed framework. In modified PROMETHEE-II technique the AHP technique has been used for calculating weights of the alternatives. The QoE parameters considered for the evaluation of this technique are time to finish, RAM requirement and storage requirement, number of cores, uplink and downlink latency, and uplink and downlink bandwidth. The working of modified PROMETHEE-II technique is explained in Algorithm 3. As per Algorithm 3 Firstly, we create a matrix of $m \times n$ between Fog environment $FE = FE_1, FE_2, \dots, FE_m$ and QoE parameters $P = P_1, P_2, \dots, P_n$. The data containing all the alternative and the criteria are better described in form of a table with $m \times n$ evaluations. Every row defines the alternatives which are Fog Environment and the criteria which are the QoE parameters as shown in the matrix in Eq3.

$$FE_{m \times n} = \begin{matrix} FE_1 \\ \cdot \\ \cdot \\ FE_m \end{matrix} \begin{bmatrix} P_{01} & \cdot & \cdot & P_{0n} \\ P_{11} & \cdot & \cdot & P_{1n} \\ \cdot & \cdot & \cdot & \cdot \\ P_{m1} & \cdot & \cdot & P_{mn} \end{bmatrix} \dots \dots \quad (3)$$

In the next step, the preference value is calculated. We assume $P_j(a)$ as the value of a criteria j for FE_a . The difference of value of a criteria j for two decisions a and b is noted as $d(FE_a, FE_b)$ which is given in Eq4. Also, $P_j(FE_a, FE_b)$ represents the preference value of a criteria j for two decisions FE_a and FE_b . The preference functions used to compute these preference values is represented in Eq5. After calculating the preference values the next step is to evaluate the weights of the parameters. Weight calculation of parameter is done with the help of analytical hierarchical process (AHP) technique in the next step of Algorithm 3. After this step, the preference function is calculated with the help of parameter weights. Let C_r is set of criteria and W_j is the weight associated to the criteria j . The global preference index for decisions a and b is defined in Eq6. Lastly, the evaluation of positive and negative outranking flow is done. For this, every decision a , the positive outranking flow $\Phi^+(FE_a)$ and negative outranking flow $\Phi^-(FE_a)$ is computed. Let A be the set of possible decisions and m is defined as the number of decisions. The positive outranking flow is evaluated by the Eq7 for decision a and negative outranking flow is evaluated by Eq8 for decision a . In the end, the total outranking flow $\Phi(FE_a)$ for a possible decision is computed with the help of Eq9. The

higher the value of decision, the better it is considered. In our FE selection area where we are calculating the ranks for best allocation, we will select the maximum outranking flow of any particular decision.

$$d(FE_a, FE_b) = P_j(FE_a) - P_j(FE_b) \tag{4}$$

$$p_j(FE_a, FE_b) = F(d(FE_a, FE_b) \text{ with } \forall x \in \varepsilon) \text{ } -\infty, \infty [0 \leq F(x) \leq 1] \tag{5}$$

$$\prod(FE_a, FE_b) = \sum_{j \in P} W_j * P_j(FE_a, FE_b) \tag{6}$$

$$\Phi^+(FE_a) = \sum_{x \in A} \pi(FE_a, x) \tag{7}$$

$$\Phi^-(FE_a) = \sum_{x \in A} \pi(FE_a, x) \tag{8}$$

$$\Phi(FE_a) = \Phi^+(FE_a) - \Phi^-(FE_a) \tag{9}$$

Algorithm 3: Modified POMETHEE-II Technique

Input: FE, Attributes

Output: Best possible alternative

Begin

Create a matrix between the FE and the attributes (m*n).

The preference value is calculated (FE_a, FE_b).

Weight evaluation is done by AHP method.

A Global function and global preference index is evaluated.

For every decision positive $\Phi^+(FE_a)$ and negative $\Phi^-(FE_a)$ outranking flow is evaluated

Total outranking flow is evaluated for evaluating the best possible attribute.

End

3.2.4 Cost Mapping Stage

In the third stage of the proposed framework, the Latency_Mapp[] buffer is provided as an input for evaluation. All the three stages are connected with each other as output of one stage acts as an input to the other stage. In Algorithm 4 for all the values of Latency_Mapp[], buffer cost is evaluated in the next step using Eq 10 and stored in Cost_Mapp[] buffer. Here, Algorithm 3 is applied on the Cost_Mapp[] buffer for the allocation of FE to the smart application by satisfying the user cost requirements. Also, Application run time is evaluated using Eq12, operational cost is calculated using Eq16. Table4 represents all the symbols used for the evaluation of Algorithms. After this evaluation, the values are stored in FE4[] and top n Fog environment values are selected and rest of the Fog environment values are discarded. If FE4[] first value is 1, then the FE is allocated to the SA. After this process of allocating the FE to SA, the allocated FE are stored in Alloc_FE[]. Otherwise, the rest of the FE are stored in WL[].

Symbol	Representation
q	Charge per minute for F.S.P
t _a	Application run time for F.S.P
η	Number of Tasks

η	Average runtime of tasks
ρ	Operational Cost
λ	Time to finish
m	Number of cores
a	Applications
ρ_c	Cost per core

Table 4: Symbols used in the Equations

$$\rho = g * m * t_a \quad (10)$$

$$\rho_c = (g * \eta * \eta_r) / m \quad (11)$$

$$t_a = (\eta * \eta_r) / m \quad (12)$$

$$m = (\eta * \eta_r) / t_a \quad (13)$$

$$\text{Cost} = g * \eta * \eta_r \quad (14)$$

$$\text{Where } t_a = \eta * \eta_r \quad (15)$$

$$\text{Operational cost} = g * t_a \quad (16)$$

Algorithm 4: Cost Mapp Algorithm

Input: Latency_Mapp[]

Output: FE4[], FE5[]

Begin

for all values in Latency_Mapp[] **do**

Calculate $\rho = g * m * t_a$

Store ρ values in Cost_Mapp[]

for all values in Cost_Mapp[]

Apply Algorithm 3 on the Cost_Mapp[] values

Store the ranks of the values in FE4[]

Select the top 10 low rank values from FE4[]

Store these values in FE5[]

Discard the rest of the values

if FE5[] values = 1

Allocate the FE with rank 1 to the SA

else

Discard the remaining values in FE5[]

end if

end for

end for

End

4 Experimental Work

This section gives a detailed overview of evaluation of the proposed framework on the basis of nine QoE criteria's [Subbaraj and Thiyagarajan, 2021] like RAM requirement, storage requirement, latency (up-link and down-link), bandwidth (up-link and down-link), number of cores, time to finish, and cost. The experimental setup is divided into three stages—Resource mapping analysis, Latency mapping analysis, and Cost mapping analysis. The main aim of this experiment is to provide allocation of Fog environment to the demanding smart applications with respective to its demands. Experimental setup is performed on a device with specifications 64bit OS, 164 GB RAM, and processor Intel (R) Core (TM) i7. The experiment for the proposed framework is divided into three stages explained in detail below:

- Resource Mapping Analysis
- Latency Mapping Analysis
- Cost Mapping Analysis

4.1 Resource Mapping Analysis

To analyse the first stage of the proposed framework Fog environment dataset attribute value, go through the Algorithm 1 as discussed in section3.2.1 for smart application 1. Suppose smart application 1 is in need of the Fog environment which has the following parameters such as minimum bandwidth requirement (3 MB), data to transfer is (40 MB), avg runtime of task is (9), total time in which to complete is (265), total core requirement is (7).

Fog Environment (F.E)	RAM (gb)	Storage (gb)	No. of cores (CPU cores)
F.E 1	7	8	50
F.E 2	8	10	43
F.E 12	6	10	45
F.E 13	3	6	19
F.E 14	6	2	63
F.E 20	6	2	43
F.E 24	6	8	57
F.E 31	8	2	41
F.E 34	2	10	58
F.E 35	7	7	37
F.E 42	4	5	24
F.E 43	2	5	71
F.E 44	6	7	78
F.E 53	6	9	32

F.E 54	3	6	42
F.E 55	5	5	50

Table 5: Filtered Fog Environment in Stage 1

For SA 1, Algorithm 1 is applied on the dataset as shown in Table3 and the filtered Fog environment values are listed in Table5. Resource mapping analysis is required to save unnecessary time taken for the allocation of Fog computing environment to smart application.

4.2 Latency Mapping Analysis

In this section the filtered Fog environment from the Resource mapping stage acts as an input for the Algorithm 2. In this stage analysis, the results obtained from the Algorithm 2 are listed in Table6 and Table 7 respectively. Here, Table6 represents the output after calculating the time to finish for the filtered Fog environments. Whereas, Table7 represents the output after applying Algorithm 3 on the Fog environment.

Fog Environment (F.E)	RAM (gb)	Storage (gb)	No. of cores (CPU cores)	uplink latency (micro sec)	downlink latency (micro sec)	uplink bandwidth (gbps)	downlink bandwidth (gbps)	Time to finish (micro sec)
F.E 1	7	8	50	555	2677	27	74	81
F.E 2	8	10	43	335	2426	85	56	75
F.E 12	6	10	45	400	2779	100	22	85
F.E 13	3	6	19	526	4048	76	6	40
F.E 14	6	2	63	643	2667	15	32	43
F.E 20	6	2	43	305	3666	44	35	67
F.E 24	6	8	57	347	3495	73	65	26
F.E 31	8	2	41	647	2694	79	93	42
F.E 34	2	10	58	535	2498	22	61	97
F.E 35	7	7	37	269	3289	86	82	41
F.E 42	4	5	24	558	3686	55	30	30
F.E 43	2	5	71	749	2357	55	77	68
F.E 44	6	7	78	338	2682	59	18	44
F.E 53	6	9	32	742	3692	50	94	61
F.E 54	3	6	42	286	2855	42	82	58
F.E 55	5	5	50	707	2672	96	16	62

Table 6: Fog Environment Parameters

Sr.	Alternative (F.E)	Positive Flow $\Phi^+(FE_a)$	Negative Flow $\Phi^-(FE_a)$	Total Flow $\Phi(FE_a)$	Rank
1	F.E 1	0.13183	0.29055	-0.15871	78
2	F.E 2	0.04467	0.35292	-0.30825	98
3	F.E 12	0.10246	0.32437	-0.22191	90
4	F.E 13	0.27077	0.07840	0.19238	13
5	F.E 14	0.17965	0.20824	-0.02860	57
6	F.E 20	0.20754	0.08619	0.12135	25
7	F.E 24	0.06132	0.26759	-0.20627	85
8	F.E 31	0.11540	0.15714	-0.04173	61
9	F.E 34	0.33888	0.05655	0.28232	4
10	F.E 35	0.31139	0.04957	0.26182	5
11	F.E 42	0.35772	0.02084	0.33688	1
12	F.E 43	0.23029	0.11702	0.11328	26
13	F.E 44	0.11938	0.32167	-0.20230	84
14	F.E 53	0.15720	0.13859	0.01861	48
15	F.E 54	0.35174	0.05286	0.29887	3
16	F.E 55	0.17417	0.12769	0.04649	43

Table 7: Fog environment Ranks

4.3 Cost Mapping Analysis

In this ,the experimental setup of Algorithm 4 is evaluated and the Fog environment are ranked as per Table7. Further, Table8 represents the attribute values after calculating cost for the Fog environment values. Table8 represents the Fog environment attributes values including cost as a new parameter in this stage. Here, Table9 represents the ranks of the values by again applying Algorithm 3 to the Fog environment new parameters. After the experimental analysis of the proposed framework the FE 1 is allocated to SA 1 by satisfying all its user demands.

Fog Environment (F.E)	RAM (gb)	Storage (gb)	No. of cores (CPU cores)	uplink latency (micro sec)	down link latency (micro sec)	uplink bandwidth (gbps)	down link bandwidth (gbps)	Time to finish (micro sec)	Cost (\$)
F.E 1	7	8	50	555	2677	27	74	81	3
F.E 2	8	10	43	335	2426	85	56	75	19

F.E 12	6	10	45	400	2779	100	22	85	21
F.E 13	3	6	19	526	4048	76	6	40	17
F.E 14	6	2	63	643	2667	15	32	43	8
F.E 20	6	2	43	305	3666	44	35	67	3
F.E 24	6	8	57	347	3495	73	65	26	18
F.E 31	8	2	41	647	2694	79	93	42	21
F.E 34	2	10	58	535	2498	22	61	97	32
F.E 35	7	7	37	269	3289	86	82	41	11
F.E 42	4	5	24	558	3686	55	30	30	20
F.E 43	2	5	71	749	2357	55	77	68	29
F.E 44	6	7	78	338	2682	59	18	44	6
F.E 53	6	9	32	742	3692	50	94	61	23
F.E 54	3	6	42	286	2855	42	82	58	4
F.E 55	5	5	50	707	2672	96	16	62	10

Table 8: Fog environment new parameter values

Sr.	Alternative (F.E)	Positive Flow $\Phi^+(FE_a)$	Negative Flow $\Phi^-(FE_a)$	Total Flow $\Phi(FE_a)$	Rank
1	F.E 1	0.13183	0.29055	-0.15871	78
2	F.E 2	0.04467	0.35292	-0.30825	98
3	F.E 12	0.10246	0.32437	-0.22191	90
4	F.E 13	0.27077	0.07840	0.19238	13
5	F.E 14	0.17965	0.20824	-0.02860	57
6	F.E 20	0.20754	0.08619	0.12135	25
7	F.E 24	0.06132	0.26759	-0.20627	85
8	F.E 31	0.11540	0.15714	-0.04173	61
9	F.E 34	0.33888	0.05655	0.28232	4
10	F.E 35	0.31139	0.04957	0.26182	5
11	F.E 42	0.35772	0.02084	0.33688	1
12	F.E 43	0.23029	0.11702	0.11328	26
13	F.E 44	0.11938	0.32167	-0.20230	84
14	F.E 53	0.15720	0.13859	0.01861	48
15	F.E 54	0.35174	0.05286	0.29887	3
16	F.E 55	0.17417	0.12769	0.04649	43

Table 9: Final Fog environment rank values

5 Performance analysis

The performance evaluation for the proposed framework is conducted by various performance metrics such as allocation time, average cost of application, average profit by the Fog environment, resource utilization, and the number of applications runs

within given latency. The QoE parameter values for RAM, storage, no. of cores, uplink latency, downlink latency, uplink bandwidth, and downlink bandwidth are randomly generated using uniform distribution. The simulation is run for 120 minutes for every application and resource considered for these metrics. For a better understanding, results of every specific metric are combined in a single 2D graph. The graphs are described in the following subsections. The time taken by the Fog environment to allocate the smart applications is defined as the allocation time. In Figure 3 (a), the allocation time of the proposed model is less as compared to the other two Fog models. The proposed model performs better because of it is divided into three stages. These stages reduce the number of Fog resources which needs to be allocated to the smart applications.

Whereas in the traditional Fog model, random allocation takes place which increase the allocation time of the Fog resources. Initially, the traditional Fog model allocates the resources to the smart applications on a first come basis which reduces the allocation time. But, after some time when the system gets overloaded, Fog resources are left underutilized and over utilized, respectively. This situation results in a sudden increase in allocation time of the Fog environment. Whereas our proposed model was able to select only focused Fog environment which are suitable for the demanding smart applications. In Figure 3, allocation time of all the Fog resources is presented compared with the other traditional and Enhanced Fog computing models respectively. In this graph, the average allocation time of the proposed framework is 2 seconds, whereas the traditional Fog model has an average time of 7 seconds. On the other hand, the enhanced Fog model is taking an average of 4 seconds. The average cost of applications is defined here by the response time of the Fog environment. The proposed Fog model is having minimum cost of the smart applications depending on the response time of the Fog environment.

In the traditional Fog model, the cost of smart applications is very high as they are not getting the demanded resources in the specified time. Average cost of applications is evaluated by using Eq13. The graph in Figure 3(c) shows the variation of cost for all the smart applications under the duration of the experiment. The profit obtained by the Fog environment is defined as the waiting time of its resources. The waiting time here is the cost of the Fog resources for the smart applications. The cost associated is where Fog environment needs to evaluate the smart application. In case the Fog environment is not able to run the smart application, then the waiting time is increased and there will be less profit by the Fog environment. In a traditional Fog environment, some latency-sensitive smart applications operate under their demanded time. But there are some smart applications which are not able to give the desired services to the users under the specific time. This situation results in wastage of all previously allocated resources.

On the other hand, the proposed model is allocating better Fog resource in less time which stops the underutilization and over utilization of the resources. Figure 3(d) represents the better profit calculated by the Fog environment in less time which results in more earnings of the Fog environment. The resource utilization of Fog resources signifies the availability of the resources for allocation of Fog environment. In Figure 3(e), the resources utilized by the proposed model is around 80%, but the traditional and enhanced Fog model utilised resources are around 70% and 84% respectively. This figure represents the utilized resources on the basis of the current resources in the computing models used. Due to high demand in application requests, the QoE-satisfied applications are decreasing in number. These are set by the number

of waiting time taken by the applications. Figure 3(b) represents the number of applications obtaining the time deadline of the customer services by a user. The graph shows that the maximum number of applications is achieving their timeline in the proposed model in comparison with the traditional and enhanced Fog model.

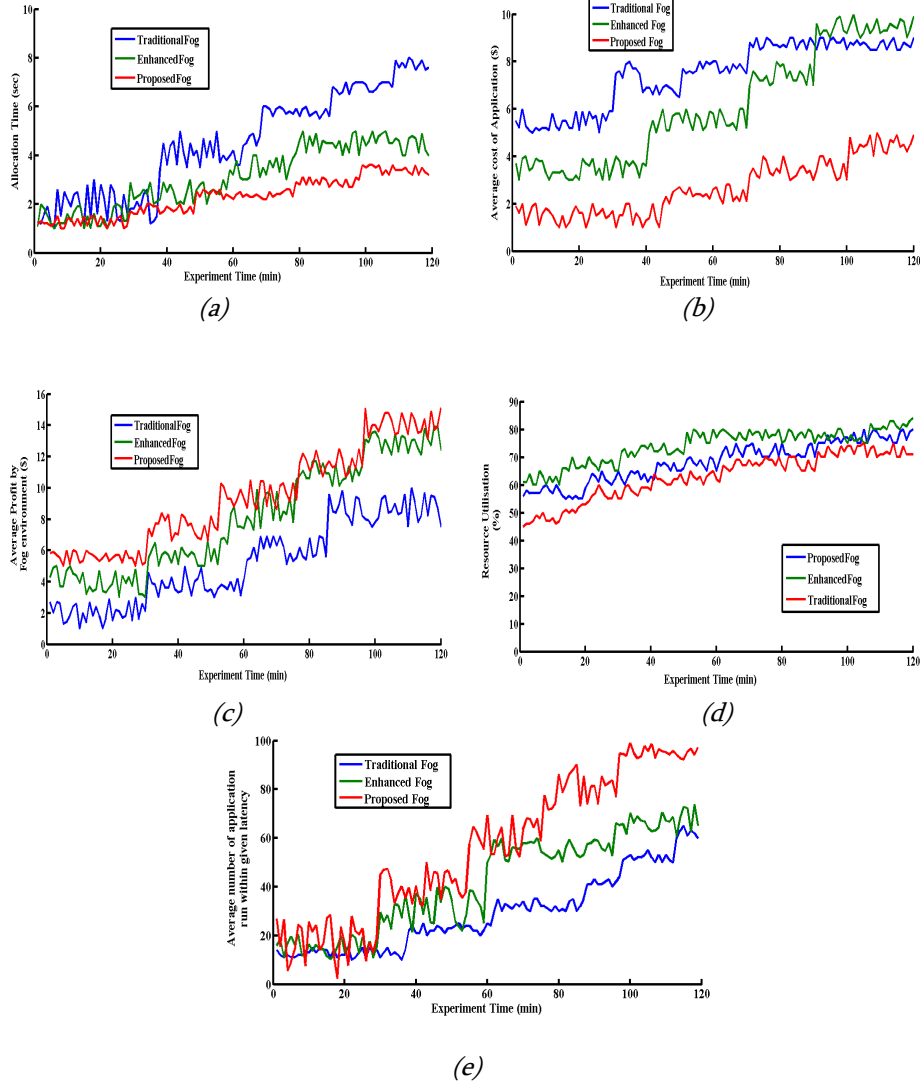


Figure 3: (a) Average Allocation time of Fog environment, (b) Average Cost of Applications, (c) Average Profit by Fog environment, (d) Resource Utilisation by Fog environment, (e) Average number of Application run within given latency

6 Conclusion

In this paper, cost effective scheduling of Fog environment is presented with the help of the MCDM technique. Various QoE parameters are considered for the evaluation of Fog environment, including RAM and storage requirements, uplink and downlink latency, uplink and downlink bandwidth, number of cores required, time to finish and cost. The proposed framework is divided into three phases for better allocation of Fog environment to the smart application. Every stage of the proposed framework filters the Fog environment by applying Algorithm 1, Algorithm 2, Algorithm 3, and Algorithm 4, respectively. For better execution, scheduling integration of two MCDM techniques has been used. The proposed framework is validated by creating an experimental setup which consists of three analysis stages. A synthetic dataset has been generated which helps in better generation of QoE attribute values. Later, the proposed framework is compared with the help of other two Fog models which are traditional model and enhanced model. The comparison of the proposed framework with the other two is possible with the help of various metrics including average allocation time, average cost of applications, average profit by the Fog environment, resource utilization, and number of applications run within the given latency. The experimental result shows that the proposed framework has a better performance in terms of all five metrics.

Acknowledgements

The authors are thankful to the organisation Jaypee University of Information Technology, Wakhnaghat, Solan (H.P) for providing the research facilities and lab for executing the experiment.

References

- [Afrin et al., 2017] Afrin, M., Razzaque, M., Anjum, I., Hassan, M. M., Alamri, A., et al. (2017). Tradeoff between user quality-of-experience and service provider profit in 5g cloud radio access network. *Sustainability*, 9(11):2127.
- [Brans et al., 1986] Brans, J.-P., Vincke, P., and Mareschal, B. (1986). How to select and how to rank projects: The promethee method. *European journal of operational research*, 24(2):228–238.
- [Fan et al., 2017] Fan, J., Wei, X., Wang, T., Lan, T., and Subramaniam, S. (2017). Deadline-aware task scheduling in a tiered iot infrastructure. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pages 1–7. IEEE.
- [Goudarzi et al., 2022] Goudarzi, M., Palaniswami, M., and Buyya, R. (2022). Scheduling iot applications in edge and fog computing environments: A taxonomy and future directions. *arXiv preprint arXiv:2204.12580*.
- [Hosseini et al., 2022] Hosseini, E., Nickray, M., and Ghanbari, S. (2022). Optimized task scheduling for cost-latency trade-off in mobile fog computing using fuzzy analytical hierarchy process. *Computer Networks*, 206:108752.
- [Hussain and Merigó, 2022] Hussain, W. and Merigó, J. M. (2022). Centralised quality of experience and service framework using promethee-ii for cloud provider selection. In *Intelligent processing practices and tools for e-commerce data, information, and knowledge*, pages 79–94. Springer.

- [Khansoltani et al., 2022] Khansoltani, A., Jamali, S., and Fotohi, R. (2022). A request redirection algorithm in content delivery network: Using promethee approach. *Wireless Personal Communications*, 126(2):1145–1175.
- [Kiani and Ansari, 2017] Kiani, A. and Ansari, N. (2017). Toward hierarchical mobile edge computing: An auction-based profit maximization approach. *IEEE Internet of Things Journal*, 4(6):2082–2091.
- [Liu et al., 2018] Liu, C., Li, K., and Li, K. (2018). Minimal cost server configuration for meeting time-varying resource demands in cloud centers. *IEEE Transactions on Parallel and Distributed Systems*, 29(11):2503–2513.
- [Mahmud et al., 2020] Mahmud, R., Srirama, S. N., Ramamohanarao, K., and Buyya, R. (2020). Profit-aware application placement for integrated fog–cloud computing environments. *Journal of Parallel and Distributed Computing*, 135:177–190.
- [Mann, 2022] Mann, Z. A. (2022). Decentralized application placement in fog computing. *IEEE Transactions on Parallel and Distributed Systems*.
- [Martinez et al., 2020] Martinez, I., Hafid, A. S., and Jarray, A. (2020). Design, resource management, and evaluation of fog computing systems: a survey. *IEEE Internet of Things Journal*, 8(4):2494–2516.
- [Mishra et al., 2019] Mishra, M. K., Ray, N. K., Swain, A. R., Mund, G. B., and Mishra, B. S. P. (2019). An adaptive model for resource selection and allocation in fog computing environment. *Computers & Electrical Engineering*, 77:217–229.
- [Ni et al., 2017] Ni, L., Zhang, J., Jiang, C., Yan, C., and Yu, K. (2017). Resource allocation strategy in fog computing based on priced timed petri nets. *IEEE Internet of Things Journal*, 4(5):1216–1228.
- [Pham et al., 2017] Pham, X.-Q., Man, N. D., Tri, N. D. T., Thai, N. Q., and Huh, E.-N. (2017). A cost-and performance-effective approach for task scheduling based on collaboration between cloud and fog computing. *International Journal of Distributed Sensor Networks*, 13(11):1550147717742073.
- [Sandhu and Sood, 2015] Sandhu, R. and Sood, S. K. (2015). Scheduling of big data applications on distributed cloud based on qos parameters. *Cluster Computing*, 18(2):817–828.
- [Sidhu and Singh, 2019] Sidhu, J. and Singh, S. (2019). Using the improved promethee for selection of trustworthy cloud database servers. *Int. Arab J. Inf. Technol.*, 16(2):194–202.
- [Singh et al., 2020] Singh, M., Baranwal, G., and Tripathi, A. K. (2020). Qos-aware selection of iot-based service. *Arabian Journal for Science and Engineering*, 45(12):10033–10050.
- [Subbaraj and Thiyagarajan, 2021] Subbaraj, S. and Thiyagarajan, R. (2021). Performance oriented task-resource mapping and scheduling in fog computing environment. *Cognitive Systems Research*, 70:40–50.
- [Yao et al., 2017] Yao, H., Bai, C., Xiong, M., Zeng, D., and Fu, Z. (2017). Heterogeneous cloudlet deployment and user-cloudlet association toward cost effective fog computing. *Concurrency and Computation: Practice and Experience*, 29(16):e3975.
- [Yu et al., 2017] Yu, L., Jiang, T., and Zou, Y. (2017). Fog-assisted operational cost reduction for cloud data centers. *IEEE Access*, 5:13578–13586.