Mobile Networks and Applications (2020) 25:1816–1832
https://doi.org/10.1007/s11036-020-01563-x

QoE Aware IoT Application Placement in Fog Computing Using Modified-TOPSIS

Gaurav Baranwal[1] · Ravi Yadav[1] · Deo Prakash Vidyarthi[2]

Published online: 6 August 2020

Abstract
Over the years, fog computing has emerged as a paradigm to complement the cloud computing in handling the delay sensitive IoT applications in a better manner. Using fog resources, better performance such as in-time service delivery, reduced network load, optimal energy usage etc. can be achieved. With such performance gain, users availing the IoT services are more satisfied. A well-known metric Quality of Experience (QoE), used to measure the satisfaction of IoT users, can be improved by enhancing the performance of the IoT applications. Fog computing is a geographically distributed paradigm and primary service of fog computing may not include the execution of offloaded tasks/applications from the IoT devices. This makes QoE aware placement of applications in fog computing a greater challenge. Since placement algorithm is itself a computational task and both IoT applications and fog nodes need a mediator fog node to execute the placement algorithm, the placement policy should be light weighted in terms of computational complexity. This work proposes a lightweight QoE aware application placement policy in fog computing using Modified TOPSIS that prioritizes the applications and fog instances based on their expectation and computational capability respectively for the placement. Modified TOPSIS inherits all the features of classical TOPSIS while it removes rank reversal problem of classical TOPSIS. Simulation experiments, for a comparative study, depict that the proposed model not only achieves the desired resource utilization, processing time, and reduced network congestion but reduces the application placement time also significantly compared to the state of art.

Keywords Internet of Things (IoT) · Fog-integrated Cloud · Modified TOPSIS (M-TOPSIS) · Quality of Experience (QoE) · Application placement

1 Introduction

The vision of the Internet of Things (IoT) is the added automation and smartness to human life. A rapid growth in the number of IoT devices is not unforeseen. Early IoT stages have witnessed an enormous increase in data generation by the IoT devices which are further expected to grow exponentially. Cloud resources may not suffice to process such voluminous data in a timely manner [1]. It necessitates the need of local computing devices to support the requirements of IoT. The emergence of the fog computing is essentially to fill this gap [2]. Placement of IoT applications on fog resources, improves the performance such as in-time service delivery, reduced network load optimal energy usage etc. Fog computing is a supplement to cloud computing as it provides an additional layer of computing infrastructure between IoT devices and the cloud. Such integrated infrastructure is referred as fog-integrated cloud [3].

IoT applications are placed on fog nodes in anticipation of satisfactory services which can be measured using a customer centric measurement, Quality of Experience (QoE). Basically, QoE is the evaluation of the requirement and expectation of an application for a particular service. But fog computing is geographically distributed paradigm and generally the primary service of the fog nodes does not include the execution of the offloaded tasks or applications from the IoT devices. For example, primary service of router (a possible fog node) is to

✉ Gaurav Baranwal
gaurav.baranwal@bhu.ac.in

Ravi Yadav
ravics85@gmail.com

Deo Prakash Vidyarthi
dpv@mail.jnu.ac.in

[1] Department of Computer Science, Banaras Hindu University, Varanasi, India

[2] School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, India

forward data packets. Because of this, QoE dominating factors such as availability of fog nodes, processing time of fog nodes and round trip time of network change very frequently. In addition, QoE is more of a subjective matter and involves the experience of the users. This makes the design of QoE aware resource allocation policies a big challenge in fog computing. In [4], authors define QoE as degree of delight of the user of a service and it is affected by the influencing factors which are responsible for technically produced quality of the service. Thus, QoE can be improved automatically by improving its influencing factors. In an application placement, these influencing factors are network quality, resource utilization, processing time etc. By improving these influential factors, we do not only improve QoE but also make customer more loyal resulting in the reduction of the relinquish rate for the service [2]. In application placement, requirement of applications and resource capabilities of fog nodes are submitted to a mediator fog node which executes the placement algorithm and maps the applications on suitable fog nodes. As such, application placement algorithm should exhibit minimal computational complexity as the computing resources of a mediator fog node may be constrained and the network of fog nodes may be highly unstable. Therefore, in case of failure of selected mediator fog node, next node should be quickly assigned this job (need to rerun the application placement algorithm).

Various feedback based methods, such as Net Promoter Score (NPS) and Mean Opinion Scores (MOS), are proposed in [5, 6] to measure the QoE. These methods require human intervention, often defeating the very purpose of automation in IoT. Some other proposed works, to measure the QoE, are based on prediction [7, 8] but highly unstable fog environment does not seem suitable for prediction based method. A possible approach is to consider parameters which are directly related to influencing factors of QoE during the placement of IoT applications onto the fog nodes i.e. to design a QoE aware placement policy. This not only prevents the degradation in QoE but also monitors it well. We may refer these parameters as 'QoE dominated factors'. The most recent work, towards this, is [2] in which authors have proposed QoE aware application placement policy in fog computing considering QoE dominated factors in decision making. The work in [2] uses fuzzy logic to prioritize the IoT applications as well as the fog computing instances on the basis of their expectation and their computational capability respectively. Mapping of these applications, on computing instances, is formulated as a single objective multi-constrained optimization problem. Therefore, some drawbacks have been identified in [2] which are as follows. It was observed that applied concept of fuzzy logic in [2], may allot same priority to applications even with different values of application dependent metric. Similarly, same priority may be assigned to fog computational instances with differential computational capability. The third and very important drawback is the very high computational complexity of the application placement policy. These drawbacks, detailed in Section 2, are the motivation for this work.

The proposed work designs a new QoE aware policy for placement of IoT application on fog nodes by removing the identified drawbacks of the state of the art work in [2]. In the proposed work, all applications for execution are submitted to a mediator fog node. It also collects the status of available fog instances. Proposed application placement policy allocates the submitted applications to suitable fog computing instances in a way such that QoE gain of the application is maximized. To prioritize the applications as well as fog computing instances, some method with less computational complexity is warranted. Multiple metrics are required to prioritize the applications (i.e. service access rate, required resources, expected processing time) and fog computing instances (i.e. Proximity, Processing speed, Availability). This necessitates a multi criteria decision making (MCDM) method. We observed that TOPSIS (Technique for Order of Preference by Similarity to Ideal Solution) is a widely accepted, simple and efficient MCDM method. During the implementation of TOPSIS, to rank the applications and fog computing instances, it is noted that the rank reversal problem prevails in TOPSIS. Therefore, inspired from the work in [9], a modification is made to the TOPSIS that handles the rank reversal problem well during the placement of applications. The computational complexity of modified TOPSIS is also very less. After prioritizing the applications and fog computing instances, applications are mapped on these computing instances in such a way that influencing QoE factors of the users, i.e. network quality, resource utilization and processing time are improved significantly. Further, computational complexity of the proposed application placement algorithm is of polynomial time leading to a significant reduction in application placement time (i.e. time taken to rank applications and fog instances and to execute proposed placement policy) is observed over the state of art work [2]. This makes the proposed work most suitable for fog computing environment.

The key contributions of the proposed work are as follows:

a) Both, applications and computing instances, are prioritized on the basis of their dependent metrics using Modified TOPSIS.
b) Unlike state of art work [2], Modified TOPSIS have less computational complexity, assigns different priority to non-similar applications and computing instances and in addition it removes rank reversal problem.
c) Applications are mapped to fog computing instances in polynomial time to maximize the QoE of the users unlike state of art work [2] which is a single objective multi-constrained optimization problem.
d) Experimental study is provided which proves that the proposed policy performs similar to state of art work [2]

in terms of network congestion, resource consumption, processing time of applications. Also, application placement time of the proposed work is much less in comparison to state of art work.

The rest of the paper is organized as follows. Section 2 briefs a discussion on the existing work in the related area identifying the drawbacks of state of art work. An overview of the Fog integrated Cloud architecture is given in Section 3. The proposed application placement policy is described in Section 4. The computational complexity, of the proposed policy, is given in Section 5. Case study, using the proposed policy, has been illustrated in Section 6. The performance evaluation of the proposed method, through simulation experiments, has been carried out in Sections 7. Section 8 concludes the work.

2 Related work and motivation

The QoE aware application placement, in fog-integrated cloud computing, is a challenging problem. Though few works [10–13], on application placement, have been proposed by the researchers, only recent and most related QoE related placement work in fog computing, are discussed here as follows.

In [2], authors have proposed QoE-aware application placement policy in fog environment. The policy aims in prioritizing an application request based on the user's expectation. In addition, the policy calculates the capability of fog instances for matching the application's resource requirement. Users' QoE is calculated in terms of utility access, resource consumption and service delivery. Placement of applications at suitable fog instances is done to maximize the users' QoE. This policy runs in a decentralized manner.

In [5], authors have proposed a methodology called MEdia FOg Resource Estimation (MeFoRE). This prioritizes the requests and fog nodes on the basis of users' relinquish rate (how frequently user gives up) and QoE measure obtained using Net Promoter Score (NPS). NPS quantifies the feedback given by the user on a scale of 0–10. The objective is to maximize the utilization of fog nodes apart from the QoS enhancement. The methodology keeps track of Service Level Agreement (SLA) violation based on poor NPS score. In order to regain the user loyalty, number of resources is increased based on the degree of SLA violation.

P. Sondarabai et al. [14] presents various mechanisms such as context-aware application placement, caching, virtualization etc. to enhance the QoE in the fog environment. The authors describe how QoE and QoS (Quality of Service) are related and different at the same time. The authors also highlight some challenges such as trustworthiness of fog nodes, handover policy, data security etc. that arise during the deployment of the fog environment.

In [15], authors considered the notion of fog colony to create micro data centers. It consists of fog cells, a small software running on the IoT devices. In a fog colony, an IoT device is designated as control node to represent its colony. The control node manages the computational resources of its colony besides exploring the resources from other colonies to meet out its computational requirement. A fog computing management system, running in the cloud, manages the fog colony and provides computational resources from cloud as and when required. Authors have formulated the application placement on virtualized fog nodes as an Integer Linear Programming problem with deadline as QoS constraint.

Cloud computing facilitates online gaming with the required hardware and software. Due to its own limitation such as user coverage, latency, network connection etc. the QoE of players are affected. To overcome this, authors in [16] have proposed lightweight system *CloudFog* by clubbing both cloud and fog nodes. This system enables fog nodes to render the game video and streaming to the nearby players. To enhance the playback continuity and to ensure the arrival of data in players' response latency, authors have proposed an adaptive video encoding strategy and a deadline driven sender buffer scheduling respectively.

In [17], authors have proposed QoS-aware policy for the placement of multi-component IoT applications on hierarchical fog infrastructure considering applications' requirements and the states of the fog nodes at the time of allocation. For the profiling of the QoS, latency and bandwidth are considered. Geographical location, access network level, type of connection and hardware and software capabilities are used to define the characteristics of the fog nodes. Authors have developed a Java based tool, called FogTorch, by incorporating the proposed policy.

2.1 The motivation

The most recent QoE aware placement scheme is proposed in [2]. We have identified some drawbacks (DBs) of [2] and have listed them as follows. The proposed work addresses these drawbacks and presents a new application placement policy.

DB1 Authors have proposed fuzzy logic based approach in [2] to prioritize the application. The rating of the application is based on the basis of application dependent metrics namely Access Rate (*Ar*), Required Resources (*Rr*), and Processing Time (*Pt*). Fuzzy logic is used to prioritize the fog computing instances also by calculating the capacity score of the fog nodes (similar to rating of application) on the basis of computing instance dependent metrics namely Round Trip Time (*RTT*), Resource Availability (*Ra*), and Processing Speed (*Ps*). The applied fuzzy logic based approaches require maximum

and minimum values of each metric in [2], set by the mediator fog node according to the scope of the metric in the fog environment as given in Table 1.

It is also observed that in some of the cases the applications, though differing in expectations, receive the same rating. A possible reason may be that fuzzy logic based approach gives higher weightage to comparatively rigid values of parameters. Similar drawbacks are identified with the fog computing instances as well. One such case with three applications and six fog computing instances is given in Tables 2 and 3 respectively. In both these tables, rating is calculated using the methods given in [2] i.e. based on fuzzy logic.

Last column, in both the tables, shows the calculated rating which indicates that even with different expectations (i.e. different values of application metrics), applications receive the same rating. Also, fog computing instances with different capacity (different values of computing instance metrics) receive same capacity score or rating.

DB2 Fuzzy inference engine, in itself, is a complex computational job which makes the use of fuzzy based approaches computationally high. It is to be noted that the primary activity of the fog devices is to execute its own set of tasks and not to run users' application. These devices are often not equipped with high computing power. Therefore, some lightweight policy to prioritize the applications and fog computing instances will be highly appreciated.

DB3 Application placement problem has been formulated as an Integer Linear Problem in [2]. Some fog nodes, acting as mediator, run the application placement algorithm. As the size of the problem i.e. the number of applications and the number of computing instances grows, more computing power is needed to solve this problem in a limited amount of time. This warrants an application mapping method that runs in polynomial time and produces the result quickly.

The work, in this paper, proposes an application placement policy considering application dependent metrics such as service access rate needed for the application, resources required by the application and expected processing time of the application. The work also considers fog computing instance dependent metrics such as proximity in terms of round trip time, processing speed, and the resource availability of the fog nodes. Both, applications and computing instances, are prioritized on the basis of their dependent metrics using Modified TOPSIS. In the proposed work, non-similar applications and computing instances are assigned different priority. In fog computing, except the work in [2], no other work maximizes the QoE gain. However, the computational complexity of the method in [2] is very high. The proposed work maximizes the QoE gain with better computational complexity compared to [2]. This makes it suitable for fog computing as fog nodes have limited computing power.

Table 1 Scope of computing instance metrics and application metrics

Computing Instance Metrics		Application Metrics	
Metric	Value	Metric	Value
$[min_{Rtt}; max_{Rtt}]$	[100; 600] ms	$[min_{Ar}; max_{Ar}]$	[2; 10] per sec
$[min_{Ra}; max_{Ra}]$	[1; 10] CPU cores	$[min_{Rr}; max_{Rr}]$	[1; 8] CPU cores
$[min_{Ps}; max_{Ps}]$	[10; 70] TIPS	$[min_{Pt}; max_{Pt}]$	[30; 120] ms

3 The system architecture

As the proposed model is an incremental improvement over the one given in [2], the same architecture has been considered in the proposed work and described here for better understanding of the proposed policy. IoT applications are divided into two modules: Client Module and Main Module. Client Module helps to find the expectation of the user of the application i.e. user's preference and contextual information and sends generated results from application module to users. Main module involves data analysis, computation of decision etc. Client module of an IoT application may execute over the IoT device itself or on some closer fog nodes. Main module requires heavier computing in compare to client module. The objective of the proposed work is to identify some suitable fog nodes for the placement of the main module.

In a fog enabled IoT system, most of the works consider three tier architecture: Cloud at the top tier, fog/edge node at the middle tier, and user's devices at the bottom. Middle tier may consist of multiple layers of fog nodes. This work considers two layers in fog: Fog Gateway Nodes (FGN) at lower level close to IoT devices and Fog Computational Nodes (FCNs) at higher level [2]. Three tier architecture of the fog enabled IoT system is given in Fig. 1.

The role of both types of fog nodes is summarized as follows.

Fog Computational Node (FCN) FCN provides computational services i.e. processing of tasks and analysis of data for the IoT devices. It comprises of three components: Computing component, Communication component and Controller. The architecture of FCN is given in Fig. 2. Computing component, equipped with computational resources, is responsible for the execution of the main module of the application. In fog computing, Micro Computing Instances (MCIs) are created over fog infrastructure using virtualization [18]. MCIs can be dynamically provisioned and de-provisioned as per the requirement without degrading the QoS. Communication component provides the required communication of FCNs with FGNs and IoT devices. The responsibility of the controller component is to manage all activities of computing component and

Table 2 Values of application metrics

Application (App)	Access Rate (Ar)	Required Resources (Rr)	Processing Time (Pt)	Rating (RA)
App1	2	2	120	5.6667
App2	3	3	70	5.6667
App3	3	7	50	5.6667

communication component. Controller component runs the Computing Instance Rating Unit that collects status from MCIs and calculates the rating using the method proposed in this work. Controller component sends the rating of MCIs using RESTful services to the connected FGN.

Fog Gateway Node (FGN) Computing devices, with networking in proximity of IoT applications, can be used as FGNs such as cable modems, set top boxes etc. [2]. The architecture of FGN is given in Fig. 3. IoT devices/users and FCNs, both need to subscribe/register on the FGN in order to avail the services and to offer the services respectively of fog computing. Application Initiation Unit in FGN initiates the client module of applications from the connected IoT devices. It collects the expectation of an application and supplies it to the Application Rating Unit which calculates the rating of applications using the proposed method in this work. These ratings are forwarded to the Application Placement Unit. The calculated rating of computing instances by Computing Instance Rating Unit of connected FCNs to FGN is also forwarded to the Application Placement Unit. Upon receiving both the inputs i.e. rating of applications and rating of computing instances, Application Placement Unit runs the proposed mapping algorithm and places the main module of the application on FCNs in such a way that QoE is maximized.

4 The proposed model

This work proposes a model for an effective mapping of the IoT application to the fog instances satisfying the QoE. The main steps, of the algorithm, are as follows. Application Rating Unit in FGN does the Rating of Applications (RoA) on the basis of application dependent metrics. Rating of fog instances (RoF) is done on the basis of fog instance dependent metrics by the Computing Instance Rating Unit in FCN. Thereafter, the placement of applications on the fog instances takes place by the Application Placement Unit in FGN. To increase the success rate, applications with relatively higher expectations should be placed on the computing instance with relatively high capacity. The notations, used in this work, are given in Table 4.

4.1 Rating of application (RoA)

IoT user i subscribes to the FGN fn and submits the desired value of application dependent metrics i.e. $E_i = \{e_i^{Ar}, e_i^{Rr}, e_i^{Pt}\}$ for its application A_i where e_i^{Ar} is expected value of access rate, e_i^{Rr} is expected value of the required resources, and e_i^{Pt} is expected value of the processing time of application A_i. These values may lie in a given range in the fog computing environment e.g. processing time of an application may be in the range $[min_{Pt}, max_{Pt}]$. If expected value of any metric of an application does not belong to the specified range, the application will be discarded and may be placed elsewhere such as on cloud VMs [2]. To maximize the success rate, higher priority is assigned to the application with relatively higher expectation i.e. access rate and required resources should be maximized while processing time should be minimized for the application. For each application A_i, rating RA_i is calculated taking E_i as input

Since applications need to be prioritized on the basis of various metrics, MCDM methods are highly useful. Various MCDM methods such as AHP (Analytical Hierarchy Process), ANP (Analytic Network Process), TOPSIS etc. are available to rank the alternatives. TOPSIS is advantageous over other MCDM methods [19] as TOPSIS creates hypothetical best alternative and hypothetical worst alternative and

Table 3 Value of computing instance metrics

Computing Instance (CI)	Round Trip Time (Rtt)	Resource Availability (Ra)	Processing speed (Ps)	Rating (RF)
CI1	150	1	40	5.0
CI2	140	1	50	5.0
CI3	150	2	40	5.0
CI4	100	2	70	5.0
CI5	110	2	60	5.0
CI6	100	2	50	5.0

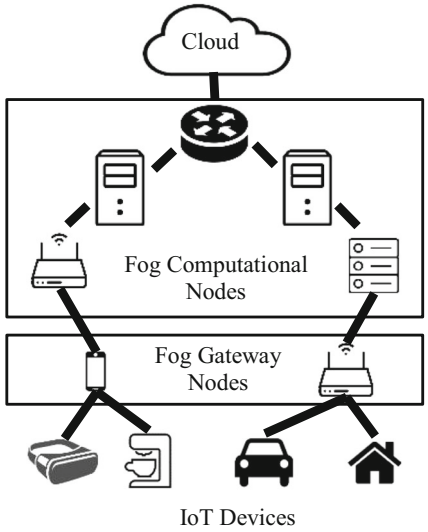

Fig. 1 Fog enabled IoT system [2]

ranks the available alternatives on the basis of their closeness from the best and the worst solutions. Further, TOPSIS is simple, easy to understand, and widely accepted MCDM method. The most important is the computational complexity of TOPSIS which is far less and helps in the design of the placement policy of less computational complexity warranted for the placement in Fog computing.

Let n be the number of applications and k the number of application dependent metrics, represented in form of matrix Q as given in Eq. (1).

$$Q = \begin{bmatrix} q_{11} & \cdots & q_{1k} \\ \vdots & \ddots & \vdots \\ q_{n1} & \cdots & q_{nk} \end{bmatrix} \tag{1}$$

In Eq. 1, q_{ij} represents the value of *jth* application dependent metrics of application i where $\{i = 1, \ldots, n\}$ and $\{j = 1, \ldots, k\}$.

The steps of the TOPSIS algorithm, applied on the matrix Q, are as follows.

Step 1: Normalize Matrix Q using Eq. (2). q'_{ij} is the element of the normalized matrix Q.

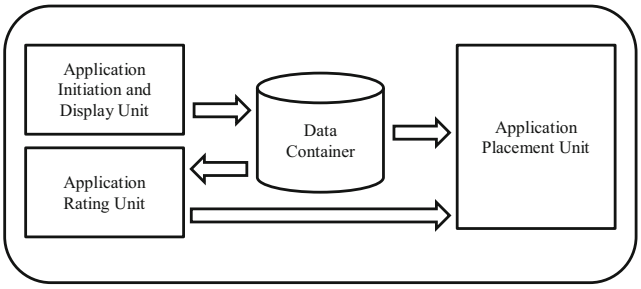

Fig. 3 Fog Gateway Node [2]

$$q'_{ij} = \frac{q_{ij}}{\sqrt{\sum_{i=1}^{n} q_{ij}^2}} \tag{2}$$

Step 2: Each metric is assigned a weight to reflect its importance. Weighted normalized matrix V is obtained by multiplying each column of matrix Q' by its corresponding weight.

$v_{ij} = w_j \times q'_{ij}$ where $\{i = 1, \ldots, n\}$, $\sum_{j=1}^{k} w_j = 1$, w_j is the weight of *jth* application dependent metric.

Since all the applications need to be compared with the same standard, in the fog environment of the proposed work, each metrics are given equal weights. As three metrics are considered, $w_1 = w_2 = w_3 = 0.333$

Step 3: Positive Ideal Solution i.e. A^+ and Negative Ideal Solution i.e. A_- are obtained from the matrix using the rules given in Eqs. (3) and (4).

$$A^+ = \left[v_1^+, \ldots, v_k^+\right] \tag{3}$$

$$A^- = \left[v_1^-, \ldots, v_k^-\right] \tag{4}$$

In this step, we try to find the best value and the worst value of each metric. A^+ represents the vector of the best value for each metric in V while A^- represents the vector of the worst value for each metric in V. Here, metrics can be of two types; one for which largest value is better (called benefit metrics), and another for which lowest value of metric is better (called cost metric). Among the considered application dependent

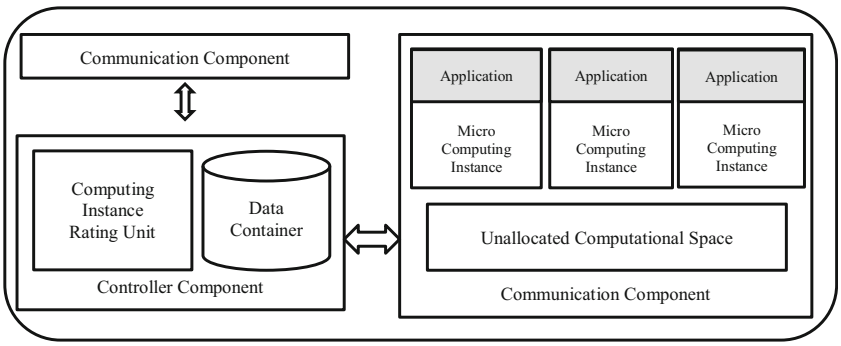

Fig. 2 Fog Computational Node [2]

Table 4 Notation

Symbol	Representation
Ar	Access Rate
Rr	Required Resources
Pt	Processing Time
Rtt	Round Trip Time
Ra	Resource Availability
Ps	Processing Speed
RA	Rating of application
RF	Rating of fog computing instance
fn	FGN
A_i	i^{th} Application
E_i	Expectation ofA_i
e_i^{Ar}	Expected value of access rate of applicationA_i
e_i^{Rr}	expected value of the required resources of applicationA_i
e_i^{Pt}	expected value of the processing time of applicationA_i
A_i	
n	Number of applications
k	Number of application dependent metrics
Q	Matrix to represent expectations of$n$ applications
q_{ij}	Value of j^{th} application dependent metrics ofA_i
w_j	Weight of j^{th} application dependent metric
O_j	Offerings of computing instance$j$
o_j^{Rtt}	Round trip time of computing instance$j$
o_j^{Ra}	Resource availability of computing instance$j$
o_j^{Ps}	Processing speed of computing instance$j$
m	Number of computing instances
l	Number of computing instance dependent metric
CQ	Matrix to represent offerings of$m$ applications
$cq_{ij}i$	Value of j^{th} computing instance dependent metric of computing instance$i$
NRR	Network Relaxation Ratio
RG	Resource Gain
$PTRR$	Processing Time Reduction Ratio
dss_ii	Data signal size of application$i$
NoA	Number of applications successfully placed

metrics, access rate and required resources are benefit metrics and processing time is cost metric.

For benefit metrics, rules to obtain the best value and the worst value are given in Eqs. (5) and (6).

$$v_j^+ = \max\{v_{ij}, i = 1, \ldots, n\} \tag{5}$$

$$v_j^- = \min\{v_{ij}, i = 1, \ldots, n\} \tag{6}$$

For cost metrics, rules to obtain the best value and the worst value are given in Eqs. (7) and (8).

$$v_j^+ = \min\{v_{ij}, i = 1, \ldots, n\} \tag{7}$$

$$v_j^- = \max\{v_{ij}, i = 1, \ldots, n\} \tag{8}$$

Step 4: Compute the distance of the requirement or expectation of each application from the positive ideal solution and the negative ideal solution i.e. S_i^+ and S_i^- respectively as given in equations (9) and (10).

$$S_i^+ = \sqrt{\sum_{j=1}^{k} \left(v_j^+ - v_{ij}\right)^2} \tag{9}$$

$$S_i^- = \sqrt{\sum_{j=1}^{k} \left(v_j^- - v_{ij}\right)^2} \tag{10}$$

Step 5: Calculate the relative closeness i.e. rating of application *RA* of each application from the ideal solution based on S_i^+ and S_i^- as given in Eq. (11).

$$RA_i = \frac{S_i^-}{S_i^+ + S_i^-} \tag{11}$$

Different metrics have different values of range. To eliminate this difference, normalization has been done in TOPSIS (Eq. (2)). However, the used normalization technique is prone to inconsistency i.e. change in the value of a particular metric for an application may change the obtained normalized values of the metric of some other applications. This may affect the rating of applications resulting in the change in their ranking. In other words, rank of applications may change if a new application is added or removed from the list. This abnormality is known as rank reversal problem. Since during mapping, applications are sorted (explained in Section 5.3), rank reversal may affect the mapping of the applications to the fog nodes eventually affecting the performance of the application placement policy.

To demonstrate rank reversal problem in the proposed work, four applications App1, App2, App3 and App4 with the values of various application dependent metrics are shown in Table 5. Fifth column represents the rating of the applications (*RA*) calculated using TOPSIS and the last column represent their ranks, obtained on the basis of the rating i.e. highest rated application to get highest rank and vice-versa.

From Table 5, it can be observed that App3 gets highest ranking. However, if App4 is removed from the list and TOPSIS is applied on the remaining applications, App3 gets lowest and App1 gets highest rating. Results are well depicted in Table 6.

From Table 6, it is clear that the removal of application reverses the order of the applications. Similarly, adding an application also may reverse the ranking of the applications. For example, in App1, App2 and App3 when App4 is added, order of App1, App2 and App3 is reversed.

A solution is warranted that can overcome the rank reversal problem but at the same time retains the basic features and benefits of TOPSIS intact. Inspired from the work in [9], it is observed that main responsible factor for the rank reversal problem is normalization (step 1 of TOPSIS algorithm) because normalized value of a metric for an application depends on the value of the same metric for all applications. Various normalization techniques, proposed in literature, are capable to remove rank reversal problem completely for a certain setting. It is identified that the range of expected values of each metrics in fog computing environment is fixed in the work in [2]. Therefore, a normalization technique that removes the rank reversal problem completely is selected in the proposed work (step 1 in modified TOPSIS algorithm in Section 4.1.1). It is because range of each metric is fixed and normalized value of various metrics depends on the range and the expected value of that metric by the application only. Therefore, with the new normalization technique, addition and removal of new applications does not affect the normalized value. Thus, in the classic TOPSIS method, normalization technique is replaced with a new normalization technique. For the placement of the applications on fog nodes, two modifications are made in the existing TOPSIS. First, equal weights are assigned to each metric and second the normalization technique is modified. The algorithm of the Modified TOPSIS is given in Section 4.1.1.

4.1.1 Modified TOPSIS to calculate RoA

The Rating of Application (RoA) is done using M-TOPSIS which is as follows.

Let k be the number of application dependent metrics for n number of applications. These are represented in form of a matrix Q as given in Eq. (12).

$$Q = \begin{bmatrix} q_{11} & \cdots & q_{1k} \\ \vdots & \ddots & \vdots \\ q_{n1} & \cdots & q_{nk} \end{bmatrix} \tag{12}$$

In matrix Q, q_{ij} represents the value of *jth* application dependent metrics of application i for $\{i = 1, \ldots, n\}$ and $\{j = 1, \ldots, k\}$. The value of k, considered in the present scenario, is 3.

The steps of M-TOPSIS (Modified-TOPSIS) algorithm, applied on matrix Q to calculate RoA, are as follows.

Table 5 TOPSIS based rating of applications

Application	Access Rate (e^{Ar})	Required Resources (e^{Rr})	Processing Time (e^{Pt})	Rating (*RA*)	Rank
App1	4	4	30	0.5185	3
App2	5	4	40	0.5223	2
App3	2	6	30	0.5323	1
App4	10	3	120	0.4449	4

Table 6 TOPSIS based rating of applications without App4

Application	Access Rate (e^{Ar})	Required Resources (e^{Rr})	Processing Time (e^{Pt})	Rating (RA)	Rank
App1	4	4	30	0.5220	1
App2	5	4	40	0.5132	2
App3	2	6	30	0.5033	3

Step 1: Calculate Normalized Matrix $Q^{'}$ by applying the following rule on the elements of matrix Q.

For benefit metrics, rule of normalization is given in Eq. (13).

$$q_{ij}^{'} = \frac{q_{ij} - min_j}{max_j - min_j} \tag{13}$$

For cost metrics, rule of normalization is given in Eq. (14).

$$q_{ij}^{'} = \frac{max_j - q_{ij}}{max_j - min_j} \tag{14}$$

Both max_j and min_j are fixed for each metric and do not depend on a particular application. Therefore, values of max_j and min_j will not change on the introduction of a new application of same type or removal of any existing application from the list.

As each metric is given equal weight and all the values are multiplied by the same factor i.e. $1/k$, step 2 of the classic TOPSIS is not required in the modified TOPSIS i.e. weighted normalized matrix is not needed $V = Q^{'}$.

Step 2: Obtain Positive Ideal Solution $A^{+} = \left[v_1^{+}, \ldots, v_k^{+}\right]$ and Negative Ideal Solution $A^{-} = \left[v_1^{-}, \ldots, v_k^{-}\right]$ using the following rules.

For benefit metrics j best value $v_j^{+} = 1$ and worst value $v_j^{-} = 0$.

For cost metrics j best value $v_j^{+} = 0$ and worst value $v_j^{-} = 1$.

Step 3: Compute the distance of the expectation of each application from positive ideal solution S_i^{+} and negative ideal solution S_i^{-} as given in equations (15) and (16)

$$S_i^{+} = \sqrt{\sum_{j=1}^{k} \left(v_j^{+} - q_{ij}^{'}\right)^2} \tag{15}$$

$$S_i^{-} = \sqrt{\sum_{j=1}^{k} \left(v_j^{-} - q_{ij}^{'}\right)^2} \tag{16}$$

Step 4: Calculate the relative closeness of each application or rating of each application i (RA_i) (from ideal solution based on S_i^{+} and S_i^{-}, as given in Eq. (17).

$$RA_i = \frac{S_i^{-}}{S_i^{+} + S_i^{-}} \tag{17}$$

Using M-TOPSIS, rating for each application is calculated and submitted to the Application Placement Unit for its allocation to suitable fog nodes.

4.2 Rating of Fog nodes (RoF)

The Controller component in FCN runs the Computing Instance Rating Unit which collects the status from MCIs and calculates the rating on the basis of its computing instance dependent metrics. For computing instance j, values of metrics is $O_j = \left\{o_j^{Rtt}, o_j^{Ra}, o_j^{Ps}\right\}$ where o_j^{Rtt} is round trip time, o_j^{Ra} is resource availability, and o_j^{Ps} is processing speed. Like application dependent metrics, computing instance dependent metrics also have numerical values in a specified range [2]. RoF of each computing instance j e.i. RF_j is calculated taking Q_j as

Table 7 TOPSIS based rating of Computing Instances

Computing Instance (CI)	Round Trip Time (o^{Rtt})	Resource Availability (o^{Ra})	Processing speed (o^{Ps})	Rating (RF)	Rank
CI1	100	3	10	0.4910	3
CI2	200	4	20	0.5298	1
CI3	100	2	20	0.4824	4
CI4	600	3	70	0.4955	2

Table 8 TOPSIS based rating of computing instances removing CI4

Computing Instance (CI)	Round Trip Time (o^{Rtt})	Resource Availability (o^{Ra})	Processing speed (o^{Ps})	Rating (RF)	Rank
CI1	100	3	10	0.4944	3
CI2	200	4	20	0.5249	2
CI3	100	2	20	0.5407	1

input. A fog computing instance, offering better metric value, should get the higher rating. To calculate the rating of each computing instances, TOPSIS can be used. Rank reversal problem, in classic TOPSIS, prevails with computing instances as shown in Tables 7 and 8.

Table 7 shows that CI3 is the lowest rated computing instance while in Table 8 CI3 becomes the highest rating instance on the removal of the instance CI4. Each metric should get equal preference for better comparison of the computing instances of same standard. Therefore, M-TOPSIS is used to calculate the rating of each computing instances also, as was done for the applications.

4.2.1 Modified TOPSIS to calculate RoF

Let l be the number of computing instance dependent metric and m be the number of computing instances, represented in form of matrix CQ as given in Eq. (18).

$$CQ = \begin{bmatrix} cq_{11} & \cdots & cq_{1l} \\ \vdots & \ddots & \vdots \\ cq_{m1} & \cdots & cq_{ml} \end{bmatrix} \tag{18}$$

In matrix CQ, cq_{ij} represents the value of jth computing instance dependent metric of computing instance i where $\{ i = 1, \ldots, m\}$ and $j = 1, \ldots, l$ Value of l, considered in the present scenario, is 3.

The steps of M-TOPSIS algorithm, applied on matrix CQ to calculate the rating of computing instances, are as follows.

Step 1: Obtain Normalized Matrix CQ' by applying the following rule on matrix CQ.

For benefit metrics, normalization method is as given in Eq. (19).

$$cq'_{ij} = \frac{cq_{ij} - min_j}{max_j - min_j} \tag{19}$$

For cost metrics, normalization method is as given in Eq. (20).

$$cq'_{ij} = \frac{max_j - q_{ij}}{max_j - min_j} \tag{20}$$

Both max_j and min_j are fixed for each metric and do not depend on a particular computing instance. Therefore, values of max_j and min_j will not change on the addition of a new application of same type or removal of any existing application from the list. Among the considered computing instance dependent metrics, round trip time is cost metric while resource availability and processing speed are benefit metrics.

Step 2: Obtain Positive Ideal Solution $CA^+ = [cv_1^+, \ldots, cv_l^+]$ and Negative Ideal Solution $CA^- = [cv_1^-, \ldots, cv_l^-]$ using the following rules.

For benefit metrics j, best value $cv_j^+ = 1$ and worst value $cv_j^- = 0$

For cost metrics j, best value $cv_j^+ = 0$ and worst value $cv_j^- = 1$

Step 3: Compute the distance of status of each computing instance from positive ideal solution CS_i^+ and negative ideal solution CS_i^- as given in equations (21) and (22) respectively.

$$CS_i^+ = \sqrt{\sum_{j=1}^{l} \left(cv_j^+ - cq'_{ij} \right)^2} \tag{21}$$

$$CS_i^- = \sqrt{\sum_{j=1}^{l} \left(cv_j^- - cq'_{ij} \right)^2} \tag{22}$$

Step 4: Calculate the relative closeness of each computing instance i.e. rating of computing instance RF_i from ideal solution based on CS_i^+ and CS_i^- as given in Eq. (23).

$$RF_i = \frac{CS_i^-}{CS_i^+ + CS_i^-} \tag{23}$$

Using M-TOPSIS, rating is calculated for each computing instances and submitted to the Application Placement Unit of connected FGN fn for mapping of applications to fog nodes.

4.3 Mapping of Applications to fog nodes

An application with higher value of RoA represents the high intensity of associated application dependent metrics while a fog instance with higher value of RoF represents the high

capacity computing instance. Therefore, for mapping of applications, first FGN *fn* sorts both RoA and RoF (all applications and all computing instances) in descending order. It then compares the requirement of first application (sorted) with computing instances in their sorted order and assigns first application to the computing instance that satisfies the application's requirement. Both, the assigned computing instance and the application are removed from their lists. *fn* repeats same steps for the application on the top position now in the list and continues until either list of applications or list of computing instances are exhausted.

5 The computational complexity of the proposed policy

In the first phase i.e. the calculation of RoA, complexity to normalize the values of various metrics of application is $O(n \times k)$ where n is the number of applications and k is the number of application metrics. Complexity of steps for creation of positive and negative ideal solution is $O(k)$. Complexity for calculation of distance of application's expectation from positive and negative ideal solution is $O(n)$ while computational complexity of the last step i.e. calculation of rating for all application is $O(n)$ Therefore, the computational complexity of first phase is $O(n \times k)$.

In the second phase i.e. calculation of RoF, complexity to normalize the values of various metrics of all computing instances of fog nodes registered on FGN is $O(m \times l)$ where m is the number of all fog computing instances and l is the number of computing instance metrics. Complexity of the steps for the creation of positive and negative ideal solution is $O(l)$. Complexity for the calculation of distance of computing instance's capacity from positive and negative ideal solution is while computational complexity of the last step i.e. calculation of rating for all computing instances is $O(m)$. Therefore, computational complexity of the second phase is $O(m \times l)$.

In the last phase i.e. Placement Policy, the complexity of sorting the application is $O(n^2)$ and the complexity of sorting the computing instances is $O(m^2)$. In the next step, for each application, all the computing instances can be scanned in the worst case to find the most suitable one. Therefore, the computational complexity of this step is $O(n \times m)$. Thus, the computational complexity of the last phase is $O(n^2 + n \times m + m^2)$.

The overall computational complexity of the proposed application policy is $(n \times k + m \times l + n^2 + n \times m + m^2)$ where n is the number of applications, m is the number of computing instances, k is the number of application dependent metrics, and l is the number of computing instance dependent metrics. If m is large, compared to other variables, the computational complexity of the proposed policy is $O(m^2)$

6 The case study

For better illustration of the proposed policy, a FGN *fn* is considered that receives request from five applications. Applications and their expected value of application dependent metrics are given in Table 9. *fn* calculates the rating of applications. Obtained rating of applications is given in the last column of Table 9. Then *fn* queries the connected fog nodes about their computing instances. Fog nodes calculate the rating of its computing instances and forwards the obtained result to *fn.* Obtained status and rating of the seven fog computing instances are listed in Table 10. All the values of metrics and range of each metric (Table 1 in Section 2.1) is taken from [2] assuming that computing instances are satisfying the minimum applications' requirement. *fn* applies the proposed placement policy with rating of applications and fog computing instances as input. Obtained result is shown in Table 11, in which each row represents the placement of an application on the computing instance.

7 The performance study

The proposed work, for the placement of applications using modified TOPSIS, is compared with the state of art work in [2]. The drawbacks of [2] have been identified and listed in Sect. 2.1. This section exhibits how the proposed work performs in comparison to [2].

7.1 Performance metrics

For a fair comparison, same performance metrics as in [2] are considered including a new metric 'Application Placement

Table 9 Expectation and rating of applications

Application	Access Rate (e^{Ar})	Required Resources (e^{Rr})	Processing Time (e^{Pt})	Rating (RA)
App1	2	2	120	0.0795
App2	5	5	70	0.5006
App3	3	3	90	0.2582
App4	7	8	60	0.7297
App5	8	3	50	0.5863

Time' defined in this work. These are described in detail as follows.

a. Network Relaxation Ratio (NRR): This metric observes the network congestion. If FGN *fn*, places the application *i* on fog computing instance *j* at time *t*, NRR can be calculated as given in Eq. (24).

$$NRR_{ij} = \frac{2}{e_i^{Ar} \times o_j^{Rtt}} \tag{24}$$

Average NRR can be written as given in Eq. (25).

$$NRR_{avj} = \frac{1}{NoA} NRR_{ij} \tag{25}$$

where *NoA* is the number of applications successfully placed on the fog computing instances.

In this, NRR > 1 for an application reflects low network congestion possibility. For example, consider an application in which expected access rate is 2 per second (it reflects that for the application intermediate delay between two data signals will be 0.5 s) and round trip time of the fog node where the application is placed is 0.3 s. Network will be free for approximately 0.35 s even after propagating the data signal from the application to the fog node and the obtained value of NRR, in this case, is 3.33.

b. Resource Gain (RG): This performance metric measures the resource consumption of a user. If FGN fn, places an application i on fog node j at time t, RG can be calculated using Eq. (26).

$$RG_{ij} = \frac{o_j^{Ra}}{e_i^{Rr}} \tag{26}$$

Average RG is as given in Eq. (27).

$$RG_{avj} = \frac{1}{NoA} RG_{ij} \tag{27}$$

Here if RG > 1, application is able to fetch the required resources. For example, if for an application expected required resources are 2 processing cores and resource availability of fog node where the application is placed is 3 cores. Obtained value of RG, in this case, will be 1.5.

c. Processing Time Reduction Ratio (PTRR): This performance metric measures the reduction in the processing time. If FGN *fn* places an application *i* on fog node *j* at time *t*, PTRR can be calculated as given in Eq. (28).

$$PTRR_{ij} = \frac{e_i^{Pt} \times o_j^{Ps}}{dss_i} \tag{28}$$

where, dss_i is the data signal size of application *i* in terms of number of instructions.

Average PTRR can be written as given in Eq. (29).

$$PTRR_{avj} = \frac{1}{NoA} PTRR_{ij} \tag{29}$$

Here if PTRR > 1, it reflects that application is processed faster than expected. For example, if for an application expected processing time is 0.12 s, data signal size is 1000 instruction and processing speed of fog node, where the application is placed, is 30 TIPS (Thousand Instruction per Second), then the obtained value of PTRR will be 3.6.

d. Application Placement Time: As discussed in Section 5, the computational complexity of the placement policy is in polynomial time and is very less. To calculate the actual time for the placement of an application, a performance metric called Application Placement Time is considered. It measures the time required for the calculation of rating of both; applications and fog nodes. It also measures the time taken in making a decision on the placement of the applications on the suitable fog nodes.

Table 10 Status and rating of computing instances

Computing Instance (CI)	Round Trip Time (o^{Rtt})	Resource Availability (o^{Ra})	Processing speed (o^{Ps})	Rating (RF)
CI1	100	3	20	0.4766
CI2	100	2	20	0.4556
CI3	200	4	40	0.5386
CI4	300	5	30	0.4611
CI5	400	6	50	0.5389
CI6	500	8	70	0.6070
CI7	500	6	60	0.5233

7.2 Simulation experiments

Fog computing is introduced to remove the latency problem. However, if the time in taking the decision to place an application is high, fog computing may become a bottleneck. Therefore, an application placement policy is called good only when application placement time is very less. Application placement time is an extremely important parameter for Fog resources as fog computing has resource constraint and fog nodes are highly unstable. The objective of this section is to show that the proposed policy outperforms the state of art work which is proved by showing that the proposed policy gets results for NRR, RG and PTRR similar to the state of art work in lesser application placement time. To simulate the fog environment, we have used a computing node Intel(r) Core(tm) i5-4460 cpu @ 3.20ghz 4 GB RAM which acts as FGN. For simulation of the proposed policy and the state of art work, Matlab is used. The proposed work is named as TQP (TOPSIS based QoE aware placement) and state of art work [2] is named FQP (Fuzzy based QoE aware placement). Workload is generated using various simulation setting [2] as given in Table 12. For an experiment, 50 simulations are done and average is displayed in the figures.

7.2.1 Experiment 1: Performance analysis

In this experiment, both the proposed work and the state of art work are analyzed on the basis of the discussed performance metrics on the data for computing instances and applications of the case study given in Section 6. The obtained values, for different performance metrics, are given in Fig. 4.

From Fig. 4, it is observed that NRR of TQP is better in comparison to FQP. For other two parameters, i.e. RG and PTRR, TQP are insignificantly less than FQP. Application placement time, in case of TQP, is significantly lower as compared to FQP. Thus, it can be concluded that value of all the performance metrics using TQP is greater than 1 while application placement time in TQP is very less in comparison to the state of art work. Hence, the proposed work exhibits significant performance gain in limited amount of time.

7.2.2 Experiment 2: Effect of number of applications on NRR

In this experiment, TQP and FQP are evaluated on the generated workload using data in Table 12 and average NRR is observed for different number of applications as given in Fig. 5. From Fig. 5, it is clear that average NRR decreases when the number of application increases. This fact is quite obvious because resources are fixed and on growing number of applications, fog instances won't suffice. However, the performance of both FQP and TQP is approximately same with respect to NRR.

Table 11 Solution of application placement

Application	Computing Instance
App1	CI1
App2	CI3
App3	CI7
App4	CI6
App5	CI5

7.2.3 Experiment 3: Effect of number of applications on RG

In this experiment, average Resource Gain (RG) is observed on varying number of applications for the generated workload. Results are shown in Fig. 6 which makes it clear that average RG decreases on an increase in the number of applications. It seems quite natural as the resources are limited and fixed and on increased volume of applications, there will be limited computing instances to serve. However, the proposed work performs almost similar to the state of the art work with respect to RG.

7.2.4 Experiment 4: Effect of number of applications on PTRR

In this experiment, average PTRR (Processing time reduction ratio) is observed on different number of applications with both TQP and FQP. It is clear from the obtained results in Fig. 7 that as the number of applications for placement increases average PTRR decreases. Reason for this is the fixed resource pool. Like NRR and RG, TQP again performs almost similar to FQP for PTRR.

7.2.5 Experiment 5: Effect of number of FCNs on NRR

To observe the effect of number of FCNs, in this experiment we have fixed the number of applications to 10 while number of FCNs varies from 10 to 50. Average NRR is observed on different number of FCNs in case of both TQP and FQP. It is

Table 12 Simulation parameters

Parameter	Value
Expectation Metrics:	
Access rate	2–10 per sec
Resource requirement	1-8 CPU cores
Processing time	30-120 ms
Status Metrics:	
Round trip time	100–600 ms
Resource availability	1-10 CPU cores
Processing speed	10-70 TIPS
Applications service delivery deadline	250–750 ms
Number of accessible FCN per FGN	4–10
Data signal size	1000–2000 instructions

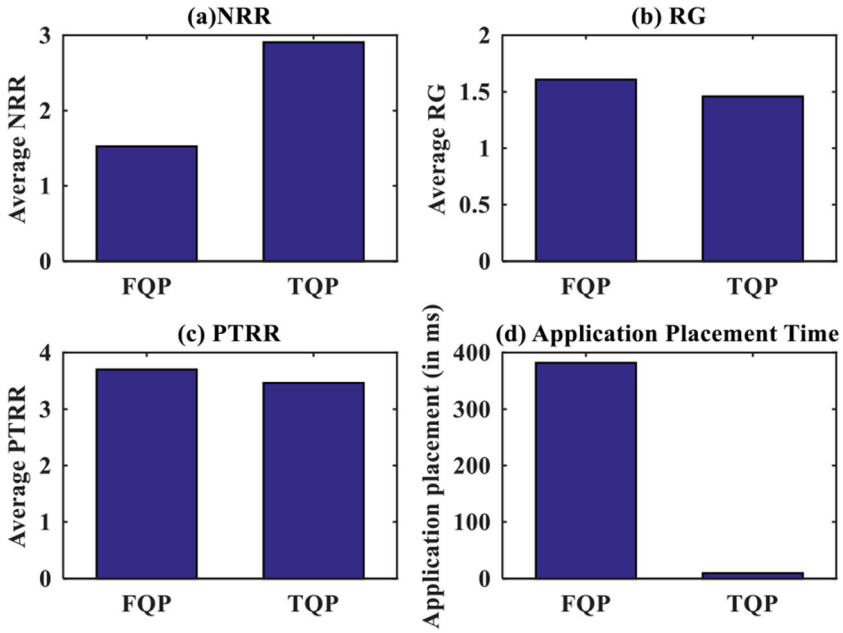

Fig. 4 Performance analysis of case study

clear, from the obtained results in Fig. 8 that Average NRR is increasing as the number of FCNs increases because number of application is fixed and they are getting better FCN with increasing number of FCNs.

7.2.6 Experiment 6: Effect of number of FCNs on RG

In the same environment as in experiment 5, Average RG is observed on different number of FCNs with both TQP and FQP and obtained results are shown in Fig. 9. It is clear that increasing number of FCNs will provide improved value of average RG in the system. Both TQP and FQP are giving nearly similar results.

7.2.7 Experiment 7: Effect of number of FCNs on PTRR

In this experiment, effect on Average PTRR for both TQP and FQP is observed on different number of FCNs in the same environment as in experiment 5 and experiment 6. Obtained results, shown in Fig. 10, depict that like NRR and RG, average PTRR can be improved by increasing the number of FCNs. Like previous experiments, TQP again performs almost similar to FQP for PTRR.

7.2.8 Experiment 8: Application placement time

This experiment observes an important parameter Application Placement Time, which is very crucial in fog computing. Observation is done by varying the number of applications and the number of fog computing instances both. Obtained

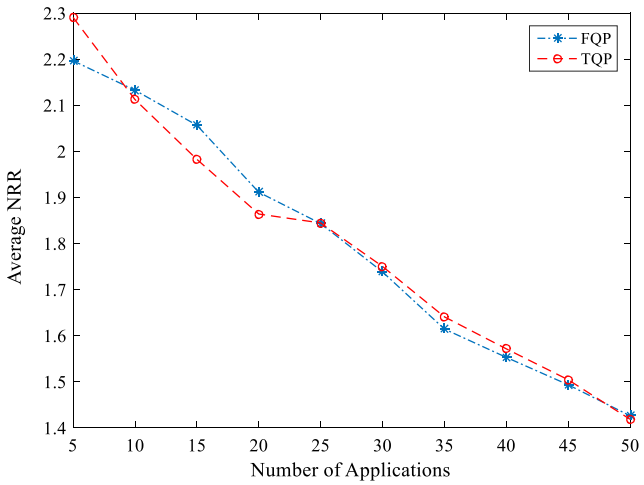

Fig. 5 NRR vs. applications

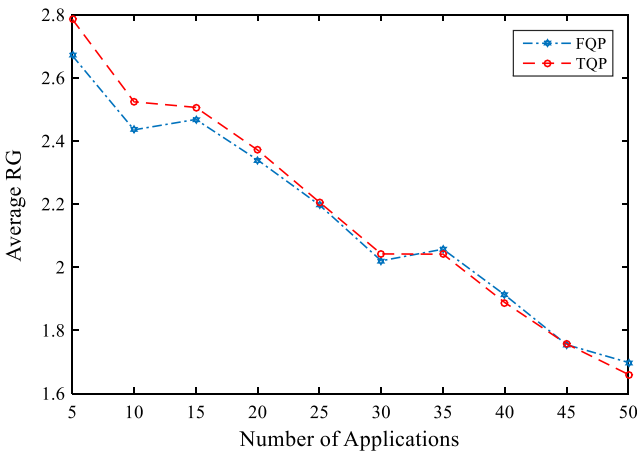

Fig. 6 Resource gain vs. applications

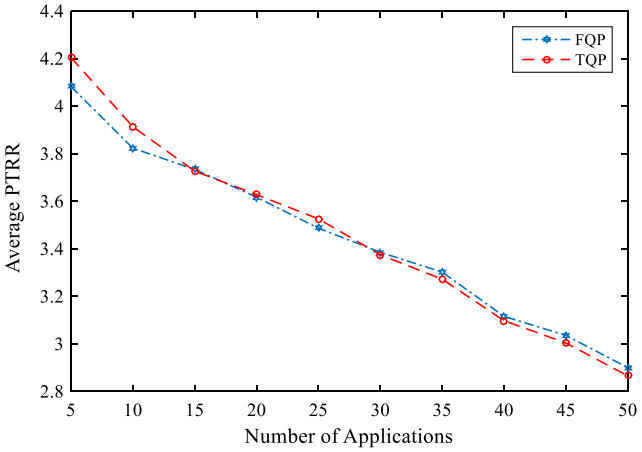

Fig. 7 PTTR vs. applications

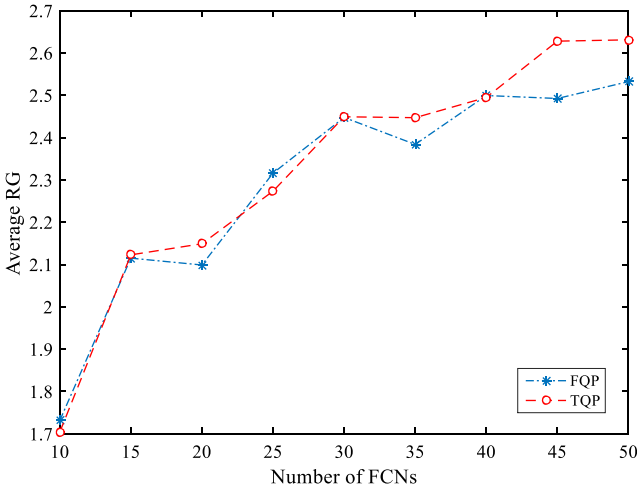

Fig. 9 Resource gain vs. FCNs

results are shown in Fig. 11 which depicts that as the size of the problem instance (number of application and number of computing instance) increases, application placement time also increases. However, with FQP increase in placement time is quite rapid while in case of TQP it is very small

7.3 Discussion on results

In most of the works related to application placement in fog computing, IoT users submit their requirement to a broker located on a fog node. This broker also receives the offering of the connected fog nodes that are ready to provide the computational resources. In the proposed work, FGN is the broker and FCN is the resource providing fog node. Broker runs the application placement algorithm to place the applications on suitable fog nodes. Application placement algorithm in itself is a computational activity and may incur high placement time, if computational complexity of the application placement algorithm is high. Many works, related to application placement, involves complex computation e.g. placement algorithm in [2] is formulated as Integer Linear Problem (ILP) using fuzzy logic concept. In [15, 20, 21], placement algorithm is formulated as ILP problem that involves complex computation; however they did not consider QoE during placement of applications. Fog computing is introduced to handle mainly delay sensitive applications effectively. If application placement time is high, it may hamper the efficient execution of delay sensitive applications losing the relevance of fog computing. Also, since primary service of the fog nodes is not solely to run application placement algorithm or to execute the offloaded applications, failure of fog nodes is inevitable. In case a FGN fails, some other FGN can overtake the load in limited time only if computational complexity of application placement i.e. application placement time is much lesser.

From the above simulation experiments it can be observed that with the proposed model, the values corresponding to all performance metrics i.e. NRR, RG and PTRR is greater than one as desired. It is also approximately equal to the values as obtained in the state of art work. The remarkable gain, of the proposed policy, is in the application placement time which is

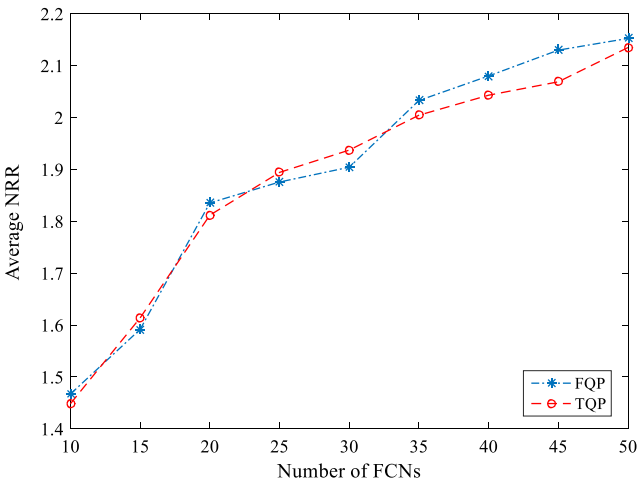

Fig. 8 NRR vs. FCNs

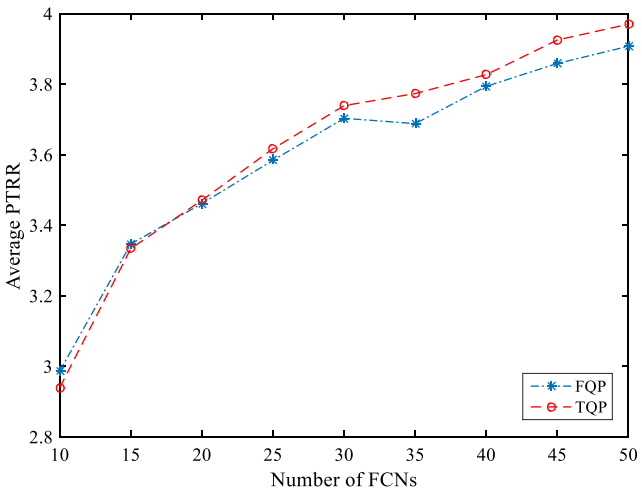

Fig. 10 PTTR vs. FCNs

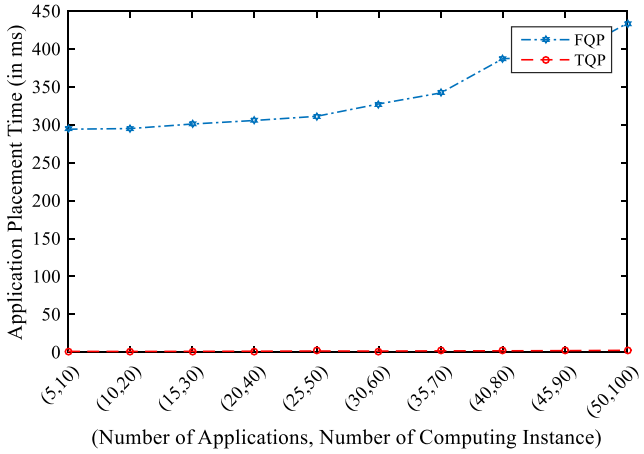

Fig. 11 Application placement time

very less as compared to the state of art. This gain also conforms to the polynomial time complexity of the proposed application placement policy given in Section 5. This makes the proposed policy most suitable in the fog computing environment which houses the fog nodes with limited computational power.

8 Conclusion

IoT devices may execute jobs/applications on the computing infrastructure of fog computing for the performance gain of the application such as in-time service delivery, reduced network load etc. An appropriate application placement policy to maximize the Quality of Experience (QoE) gain of application users is needed. Further, application placement policy in itself is a computational activity which runs on a mediator fog node and therefore should not be much complex. In this work, a QoE aware placement policy is proposed using modified TOPSIS to calculate the rating of applications and fog computing instances on the basis of the dependent metrics. The proposed policy places the applications on the basis of the calculated rating. Computational complexity of the proposed placement policy is very less in comparison to the state of the art policies. Four performance metrics i.e. Network relaxation ratio, Resource gain, Processing time reduction ratio and Application Placement Time are considered for the performance study of the proposed work in a simulated environment. It is expected that for any placement policy, the values of the first three performance metrics should be greater than one. Experiment shows that the proposed work is not only able to achieve this expectation but also is closer to the state of art work. Features that give an edge to the proposed work, are different ratings to the applications and to fog computing instances even when they have nearly same expectations and nearly same status respectively. This was not so in the state of the art work i.e. if fog nodes have values of status metrics in a small range or applications have value of expectation metrics in a small range, it assigns same rank to them. The other feature, of the proposed policy, is its reduced computational complexity as compared to the state of art work. Because of the reduced computational complexity, a significant reduction in application placement time is observed in the proposed policy in comparison to the state of art work. This makes the proposed work most suitable in fog computing environment with limited computing capacity fog resources.

As the fog owners would not share their resources without any price, an exchange based policy and price based policy may be adopted for the same. Our future work will consider developing placement policies incorporating these aspects.

Authors would like to accord their sincere thanks to the editor and the anonymous reviewers for their useful suggestions resulting in the quality improvement of this work.

References

1. Elazhary H (2019) Internet of Things (IoT), mobile cloud, cloudlet, mobile IoT, IoT cloud, fog, mobile edge, and edge emerging computing paradigms: Disambiguation and research directions. J Netw Comput Appl 128:105–140
2. Mahmud R, Srirama SN, Ramamohanarao K, Buyya R (2019) Quality of Experience (QoE)-aware placement of applications in Fog computing environments. Journal of Parallel Distributed Computing 132:190–203
3. Wang N, Varghese B, Matthaiou M, Nikolopoulos DS (2017) ENORM: A framework for edge node resource management. IEEE transactions on services computing
4. Brunnström K, Beker SA, De Moor K, Dooms A, Egger S, Garcia MN, Hossfeld T, Jumisko-Pyykkö S, Keimel C, Larabi MC, Lawlor B (2013) Qualinet white paper on definitions of quality of experience hal-00977812 f
5. Aazam M, St-Hilaire M, Lung CH, Lambadaris I (2016) MeFoRE: QoE based resource estimation at Fog to enhance QoS in IoT. In 2016 23rd IEEE International Conference on Telecommunications (ICT), pp 1–5
6. Hoβfeld T, Schatz R, Egger S (2011) SOS: The MOS is not enough!. In Third IEEE international workshop on quality of multimedia experience, pp 131–136
7. Li L, Rong M, Zhang G (2015) An internet of things QoE evaluation method based on multiple linear regression analysis. In 10th IEEE International Conference on Computer Science & Education (ICCSE), pp 925–928
8. Fiedler M, Hossfeld T, Tran-Gia P (2010) A generic quantitative relationship between quality of experience and quality of service. IEEE Netw 24(2):36–41
9. Senouci MA, Mushtaq MS, Hoceini S, Mellouk A (2016) TOPSIS-based dynamic approach for mobile network interface selection. Comput Netw 107:304–314
10. Venticinque S, Amato A (2019) A methodology for deployment of IoT application in fog. J Ambient Intell Humaniz Comput 10(5): 1955–1976
11. Casadei R, Fortino G, Pianini D, Russo W, Savaglio C, Viroli M (2019) A development approach for collective opportunistic Edge-of-Things services. Inf Sci 498:154–169

12. Yousefpour A, Ishigaki G, Gour R, Jue JP (2018) On reducing IoT service delay via fog offloading. IEEE Internet Things J 5(2):998–1010
13. Dong Y, Han C, Guo S (2018) Joint optimization of energy and QoE with fairness in Cooperative Fog Computing System. In IEEE International Conference on Networking, Architecture and Storage (NAS), pp 1–4
14. Soundarabai PB, Chellaiah PR (2018) Mechanisms Towards Enhanced Quality of Experience (QoE) in Fog Computing Environments. Springer, Fog Computing, pp 131–151
15. Skarlat O, Nardelli M, Schulte S, Dustdar S (2017) Towards qos-aware fog service placement. In 1st IEEE International conference on Fog and Edge Computing (ICFEC), pp 89–96
16. Lin Y, Shen H (2015) Cloud fog: Towards high quality of experience in cloud gaming. In 44th IEEE International Conference on Parallel Processing, pp 500–509
17. Brogi A, Forti S (2017) QoS-aware deployment of IoT applications through the fog. IEEE Internet Things J 4(5):1185–1192
18. Mahmud R, Kotagiri R, Buyya R (2018) Fog computing: A taxonomy, survey and future directions. Internet of everything. Springer, Berlin, pp 103–130
19. Behzadian M, Otaghsara SK, Yazdani M, Ignatius J (2012) A state-of the-art survey of TOPSIS applications. Expert Syst Appl 39(17): 13051–13069
20. Guerrero C, Lera I, Juiz C (2019) Evaluation and efficiency comparison of evolutionary algorithms for service placement optimization in fog architectures. Futur Gener Comput Syst 97:131–144
21. Naas MI, Parvedy PR, Boukhobza J, Lemarchand L (2017) iFogStor: an IoT data placement strategy for fog infrastructure. In 1st IEEE International Conference on Fog and Edge Computing (ICFEC), pp. 97–104