Peer-to-Peer Networking and Applications (2021) 14:1044–1057
https://doi.org/10.1007/s12083-020-01070-6

An efficient latency aware resource provisioning in cloud assisted mobile edge framework

Rajasekhar Bandapalle Mulinti[1] · M. Nagendra[1]

Received: 14 August 2020 / Accepted: 29 December 2020 / Published online: 6 February 2021

Abstract
Mobile edge computing is developing as an innovative computing paradigm that gives improved practice to mobile users through low latency connections and enlarged computation limits. As the amount of user requests is time- different, while the computation limit of the edge has is constrained, the Cloud Assisted Mobile Edge computing system is acquainted with improving the adaptability of the edge platform. To give ensured administrations at negligible framework latency, the edge resource provisioning and cloud redistributing of the cloud-assisted mobile edge computing structure ought to be wisely planned effectively. This work proposed a latency aware resource provisioning strategy for distributed cloud-assisted mobile edge computing structure. At first, the framework gets SFC requests for Virtual network functions (VNFs) to use both edge and cloud assets. Here, the efficient parameters, for example, execution time and workload of VNFs are evaluated and Fuzzy logic-based auto-scaling is executed for the overloaded VNFs that need more assets because of the progressively expanded measure of the system packets. Subsequently, the SFC requests are scheduled to the cloud-assisted edge network adequately utilizing the Adaptive Grey Wolf Optimization (AGWO) based asset provisioning algorithm. The exploratory outcomes show the superiority of the presented methodology comparing with the existing techniques as far as system cost, arrival rate, and average response time.

Keywords Resource provisioning · Optimization · Execution time · Workload measure · Autoscaling

1 Introduction

The mobile network and wireless innovation improvement have brought about different incredible mobile applications and multimedia administrations, for example, video games, face recognition, augmented reality, medicinal services, and natural language processing [1]. Furthermore, the vast majority of these applications and administrations regularly require escalated calculation and high handling, which are inconsistent with devices because of their restricted assets [2, 3]. Mobile cloud computing is viewed as a prominent solution that addresses the constraints of mobile users (MUs), in which mobile applications' intensive computations will be offloaded to incorporated clod using a remote channel to relieve the heap and broaden the battery life. In any case, high inertness is one of the fundamental deficiencies of unified distributed computing [4–6].

Moreover to the development of cloud computing, another worldview of edge computing has risen that uses assets at the edge of the system [7]. In edge computing applications and administrations are completely or somewhat served upon assets situated on the edge of the system, rather than altogether adjusted by concentrated assets in cloud server farms. Mobile edge computing (MEC) is a perfect worldview to address these issues. By conveying edge has inside the remote access organized, versatile clients can get to adequate calculation assets without experiencing the wild Internet delay. Because of the benefit of low delay, broad extensive have been given to the potential utilizations of MEC [8–12].

With the proliferation of intelligent devices, new kinds of delay-sensitive yet computation-intensive mobile applications continue rising and have drawn expanding considerations [13]. By and by, cell phones are generally

✉ Rajasekhar Bandapalle Mulinti
raajasekhar10@gmail.com

[1] Department of Computer Science and Technology, Sri Krishnadevaraya University, Anantapur, A.P., India

assets rare to help these massive computation requests. Mobile edge computing (MEC) is taken as an auspicious computing paradigm to address this issue, with the upsides of high data transmission and nearness to portable clients [14, 15]. In MEC, adequate calculation assets ought to be provisioned at a versatile edge to fulfill the QoS prerequisites. Be that as it may, the calculation assets can be under-used because of the huge temporal variety of versatile demands, bringing about over the top expense edge frameworks. The Cloud Assisted Mobile Edge computing system in [16] can well arrangement with this test [17]. A typical methodology for an asset the board in edge processing is to appoint assignments to the remote cloud or nearby servers as per a few factors, for example, energy, bandwidth consumption, having as final scope the minimization of the latency [18]. This work addresses the above issues by researching the asset provisioning issue with dynamic requests. Mobile requests are viewed as with enhanced QoS prerequisites including delay-touchy and delay-tolerant solicitations. Edge has computed all delay-sensitive and part of the delay-tolerant demands, and cloud cases are progressively rented to serve the outsourced delay-tolerant requests. The main contributions of this paper are summarized as follows,

- Effective parameters such as execution time and workload of VNFs are evaluated to enhance the resource provisioning.
- Gradually enhance the measure of packets, fuzzy logic based auto-scaling is executed for the overloaded VNFs that need more resources.
- The presented Adaptive Grey Wolf Optimization (AGWO) based resource provisioning is effectively scheduled the SFC requests to the cloud-assisted edge network.

The structure of the manuscript is sorted as Section 2 surveys the literature works concerning the proposed system. In section 3, a short discussion about the proposed framework is given, section 4 examines the exploratory results, and section 5 finishes up the paper.

2 Related work

Jingjing Guo et al. [19] proposed an On-Demand Resource Provision dependent on Load Approximation and Service Expenses in Edge Cloud Environment. The demand for assets should be evaluated ahead of time. To this end, a load estimation model dependent on the ARIMA model and BP neural system was proposed. The model can appraise the load as indicated by reported information and decrease the estimation error. Before discharging the hub assets, the client information on the hub should be relocated to other working hubs to guarantee that the client information won't be lost. Here, while choosing the movement focus on, the three measurements of load balancing, migration time utilization, and migration expenses of the cluster were measured.

Ibrahim A. Elgendy et al. [20] proposed a multiuser asset allotment and calculation offloading model with information security for mobile edge computing to address the impediments of such devices. To begin with, the computation and radio assets were mutually considered for multiuser situations to ensure the proficient usage of shared assets. What's more, an AES cryptographic procedure was acquainted as a security layer to shield delicate data from digital assaults. Moreover, an incorporated model, which together thinks about security, computation offloading, and asset distribution was detailed to limit time and vitality utilization of the whole framework. At long last, an offloading calculation was created with definite procedures to decide the ideal computation offloading choice for MUs.

Jungmin Son and Rajkumar Buyya [21] proposed a unique asset provisioning calculation for VNFs to use both edge and cloud assets. Adjusting to powerfully changing system volumes, the calculation naturally distributes assets in both the edge and the cloud for VNFs. The algorithm considers the latency prerequisite of various applications in the administration work chain, which permits the inactivity of touchy applications to lessen the start to finish arrange delay by using edge assets over the cloud. They assessed the proposed calculation in the recreation condition with enormous scope web application outstanding loads and contrast and the best in class benchmark calculation.

Chunlin Li et al. [22] proposed an adaptive resource allocation technique and an information movement calculation. The expectation calculation gives the premise to the versatile asset allotment of the edge cloud cluster. The versatile asset allocation decides the asset allotment plan of the edge cloud bunch with the most minimal assistance cost. The information migration ensures the dependability of information and accomplishes bunch load adjusting. Numerous exploratory outcomes show that our recently proposed calculation can enormously improve framework execution as far as superior cost control, higher information integrity, and load balancing.

Xu Chen et al. [23] proposed a proficient Resource Allocation for On-Demand Mobile-Edge Cloud Computing. In particular, they originally considered the asset proficient calculation offloading issue for a client, to diminish the client's asset occupation by deciding its ideal correspondence and calculation asset profile with least asset occupation and in the interim fulfilling the QoS requirement. They at that point handle the basic issue of client confirmation control

for JCC resource allotment, to appropriately choose the arrangement of clients for asset request fulfillment.

Qiang Fan and Nirwan Ansari [25] introduced a cost-aware cloudlet PlAcement in moBiLe Edge computing procedure, where both the cloudlet cost and normal E2E delay were considered in the cloudlet arrangement. To take care of the issue, a Lagrangian heuristic calculation was created to accomplish the problematic arrangement. After cloudlets were set in the organization, they additionally planned a remaining task at hand assignment plan to limit the E2E delay among clients and their cloudlets by thinking about the client's versatility.

PeiYun Zhang et al. [26] introduced an online discovery model dependent on a systematic boundary search strategy called SVM-Grid, whose development depended on an SVM. SVM-Grid was utilized to enhance boundaries in SVM. Legitimate properties of a cloud framework's running information were chosen by utilizing the Pearson relationship and head segment investigation for the model. Systems of anticipating cloud blame and refreshing flaw test information bases were proposed to advance the model.

Jun Huang et al. [27] examined the multicast directing issue in the between cloud setting with K imperatives where K , 2. Not at all like the greater part of existing calculations that are too intricate to ever be applied in pragmatic situations, a novel and quick calculation for building up multicast steering tree for bury clouds was proposed. The proposed calculation uses an entropy-based cycle to total all loads into an extensive measurement and afterward utilizes it to look through a multicast tree (MT) based on the shortest path tree.

Yong Zhang et al. [29] This work examines complex weight appropriation input-output relations and gives a portrayal of anticipated MLITD under explicit fundamental requirements based on designing practice. Besides, as per the choice factors in various number fields, this work considers the advancement of BDM with anticipated MLITD and proposes a multi-mode based PSO strategy for enhancement of choice factors.

Shangce Gao et al. [30] presented a new dendritic neuron model (DNM) by considering the nonlinearity of neurotransmitters, not just for a superior comprehension of an organic neuronal framework, yet also for giving a more helpful technique to tackling common sense issues. To accomplish its better presentation for tackling issues, six learning calculations including biogeography-based improvement, molecule swarm streamlining, hereditary calculation, subterranean insect state advancement, transformative technique, and populace based gradual learning are for the first time used to prepare it.

The problem description on Cloud Assisted Mobile Edge computing is discussed in this section along with the problem definition. In the literature all VNFs are placed in a central cloud, extra delays are expected for packets to traverse through the backbone network to reach the cloud data centre before reaching the application provider. Work also Resource provisioning problem with dynamic requests.

3 Proposed methodology

This paper introduces an effective resource provisioning methodology for VNFs to utilize both edge and cloud resources. The framework gets SFC requests and at the same time, the Fuzzy logic based auto-scaling process detects the overloaded VNFs that need more assets because of the powerfully expanded measure of the network packets. Accordingly, the SFC requests are scheduled to the cloud-assisted edge network adequately utilizing AGWO based resource provisioning algorithm. The flow diagram of the proposed methodology is given in Fig. 1.

At first, the framework gets SFC requests comprising of the source, destination, and VNF chain of the application. Here, the SFC requests are represented as$SFC_R = \{R_1, R_2, R_3, \ldots. R_n\}$. When the system traffic experiencing the VNF expands, the limit in the edge resources probably won't be sufficient to process all the expanded system traffic. In this case, we have to utilize the cloud resource to create a duplicated VNF. Right now, need to use the cloud asset to make a copied VNF. This case is controlled by the execution time and workload measure and this duplication is done by the fuzzy logic-based auto-scaling by the accompanying subsections,

3.1 VNF auto scaling for edge-clouds

The auto-scaling technique is proposed to change the necessary assets naturally to the application in demand. Here, the auto-scaling is finished by utilizing Execution Time and Workload prediction and organizes the anticipated resources by processing the necessary limit through capacity through fuzzy logic-based auto-scaling.

3.1.1 Execution time calculation

The execution time is a distinction amongst the task completion times to the task submission time for the number of tasks it is specified in condition (1),

$$E_T =_t \sum_{t=1}^{n} \left(\frac{WC_t - WS_t}{I} \right) \tag{1}$$

Where, $E_T$denotes the execution time, WC_t denotes the task completion time of VNF and WS_tdenotes the task submission time in VNF, and Idenotes the number of tasks processed in VNFs.

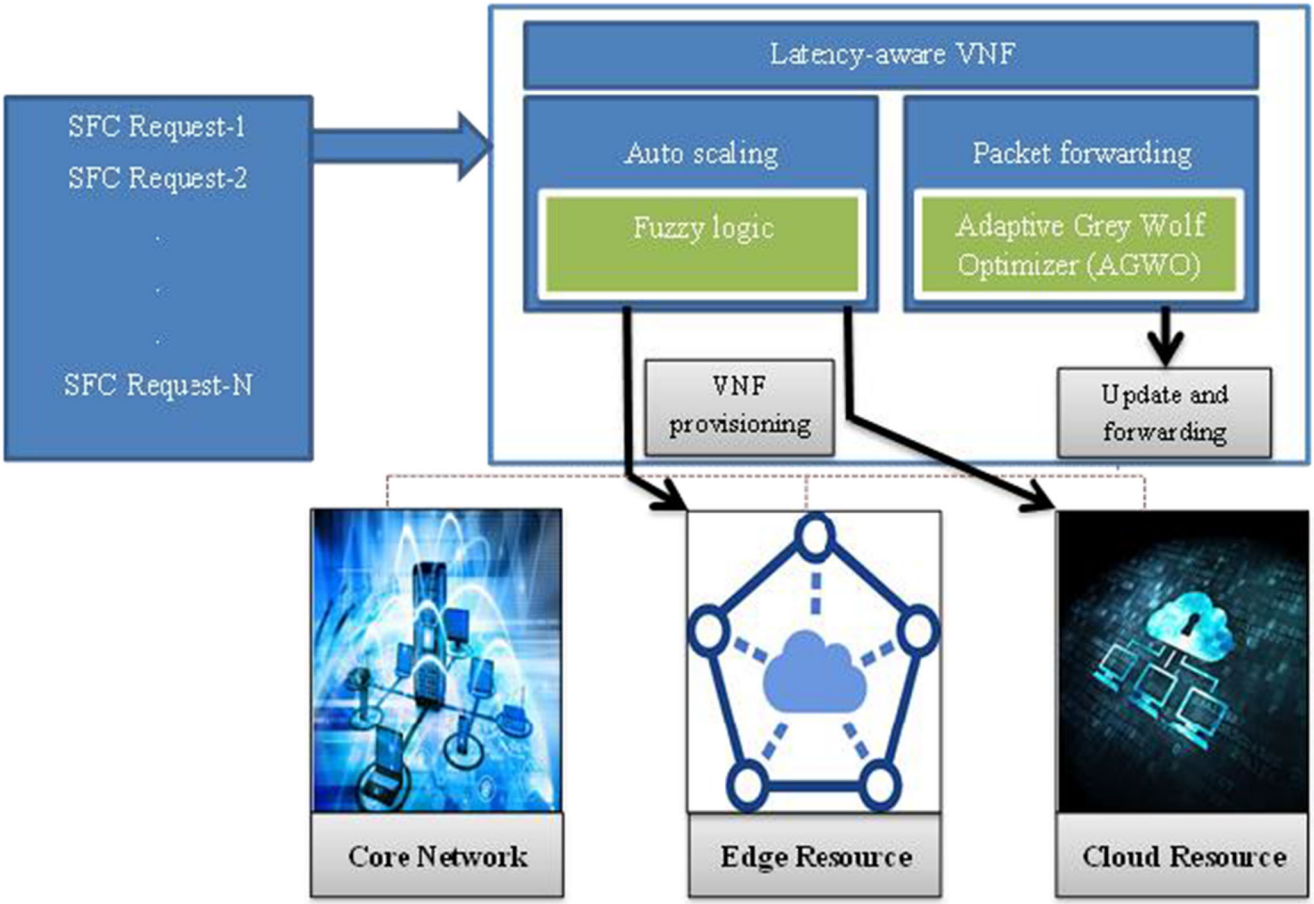

Fig. 1 Flow diagram of the proposed methodology

3.1.2 Workload calculation

The workload is the measure of processing space that the computer has been given to do in the cloud environment. It is represented in condition (2),

$$W_l\left(VNF_{cpu}\right) = \frac{\sum_{t=1}^{n} l_t\left(VNF_{cpu}\right)}{I} \tag{2}$$

Where, $W_l(VNF_{cpu})$ is the workload, I is the task. The over-utilized weight is determined utilizing condition (3) to predict the VNF limit. If the assessed weight is greater than the threshold limit, at that point the auto-scaling is processed for the VNF in any case the SFC requests sent to the streamlined resource provisioning process,

$$\widetilde{W} = \frac{E_T + W_l\left(VNF_{cpu}\right)}{N} \tag{3}$$

Here, $\widetilde{W}$ denotes an over-utilized weight, E_T denotes an execution time, $W_l(VNF_{cpu})$ denotes a workload measure, N denotes a count. These two measures are utilized for the auto-scaling of VNFs and adaptive GWO based resource provisioning in edge clouds.

3.1.3 Fuzzy logic based VNF auto-scaling

The existing VNF auto-scaling in [33]. In modified fuzzy system is a robust system where no precise inputs are required and don't need a long time to learn; it just necessities to take in the set of useful metrics and settle on the conceivable provisioning choice with fuzzy semantic guidelines for auto-scaling. Fuzzy logic has a lower learning bend when building or tuning it because of its semantic guidelines. The fuzzy logic-based VNF auto-scaling aim is to build an easy-to-use auto-scaler that does not rely on any historical data therefore the fuzzy logic-based auto-scaler is our chosen technique. Autoscaling is a cloud computing organization trait that thus incorporates or expels compute resources depending on real usage. Here the auto scaler is used to anticipate the information reliant on the execution time and workload. Our fuzzy logic-based auto-scaling uses edge assets if the assets are sufficient to give the measure of request. On account of resource outage in the edge, this technique attempts to redirect a few workloads to the cloud to disperse the load onto VNFs with enough assets. For latency-sensitive applications, we still use edge resources for meeting the necessary latency time. Less

latency-sensitive requests are diverted to VNFs put in the central cloud to use its satisfactory assets. The viable VNF auto-scaling pseudo-code for edge-clouds is given in algorithm 1.

Input Data: *VNF*: Currently running VNFs, Execution time, The work load of VNFs.
Output : Autoscaling

Begin

Read E_T and $\widetilde{W}$

Computer choice Fuzzy function F_T

$F_T = |E_T \wedge \widetilde{W}| \Big/ \alpha + |\widetilde{W}|$

Where $\wedge$ is the Fuzzy AND operation

Select the best Fuzzy match

F_T=Max{ F_T :j=1,2,3,…,N}

For each VNF v **in** *VNF* **do**

If F_T is over-utilized weight by equation (3) **then**

$loc \leftarrow$ Location of $\widetilde{W}$ (edge or cloud);

If loc.has Available Resource(v) **then**

$v \leftarrow$ duplicate VNF $\widetilde{W}$ in loc;

Else

$v \leftarrow$ duplicate VNF $\widetilde{W}$ in the cloud;

End if

Update latency map $\widetilde{W}$ to add $\widetilde{W}$;

End if

End for

End

Algorithm 1: Fuzzy logic based VNF auto-scaling and provisioning for edge-clouds

At first, the algorithm identifies the overloaded VNFs that need more resources because of the powerfully expanded amount of the network packets. The resource utilization of VNFs is continually observed and periodically detects the VNF overload. When a VNF overload is distinguished, the algorithm duplicates the VNF for load-distribution in a similar area if there are accessible assets. On the off chance that the VNF located on the edge node is overloaded, for instance, the algorithm attempts to make another VNF in the edge. On the off chance that the accessible asset in the edge is sufficient for the extra VNF, the edge node will run another VNF for the similar network function, and the network packets are sent to either VNF regardless of the application's latency necessity. In any case, for the situation, if the asset isn't sufficient in the edge, the new VNF will be put in the cloud which expands network delay. The schematic diagram of the VNF auto-scaling is given in Fig. 2.

On the off chance that the duplicated VNFs are put in various locations, the VNF forwarder considers the application necessities to choose where to forward the network packets. When the duplicated VNF is put in an alternate area, our calculation makes a system.

latency map amongst the source of the packets and the VNFs various areas to be utilized in the forwarder. By setting up the latency map at the time of VNF duplication, the forwarder can use the latency information to utilize the VNF to forward the network packet.

3.2 Resource provisioning using adaptive Grey wolf optimization (AGWO)

The issue with the existing optimization algorithms particle swarm optimization is that they have enormous time intricacy. Besides, they rely on the emphasis and the underlying

populace size, which influences their answer. On the off chance that the populace size of the cycles/age is less, at that point there is less likely to get the best arrangement. Besides, a genetic-based algorithm may give the global solution yet at an expense of high scheduling time because of the high check of cycles included while scheduling. To conquer these issues and locate a similar best solution of less time complexity, the grey wolf optimization algorithm (GWO) is utilized for resource provisioning.

The GWO has solid investigation capacity, which can dodge the calculation falling into the neighbourhood ideal. For the GWO, the correct balance between investigation capacity and misuse capacity is extremely easy to be accomplished, so it can adequately solve many convoluted issues like computational and storage complexities. Resource provisioning gives the demanded resources to the requests for their execution in a cloud-assisted edge network environment just whenever required assets are accessible in the resource pool. Grey wolf optimization is a swarm intelligent technique that imitates the administration development movement of wolves is commonplace for their group hunting. Grey wolf generally need to live in a pack and they have a firm social overwhelming hierarchy; the alpha generally in control of deciding. The Betas (β) are subordinate wolves which help the alpha in essential authority [24]. In AGWO is based on Eq. (12) numerical portrayal, the fitness solution is known as the alpha (α). The second and third most excellent solutions are named β and δ independently. The pseudo-code of adaptive grey wolf optimization is given in algorithm 2.

Input : Initialize the SFC requests as initial grey wolfs (search agents) $Y_i(i = 1,2,3\ldots n)$ and vectors $\vec{a}$, $\vec{A}$ and $\vec{C}$

Output : Latency aware Provisioning of Tasks to the system resources

For

Each Cycle $\hat{C} = 1$

repeat

for $i = 1 : Y_s$ (grey wolf pack size)

update the location of the present hunt agent utilizing condition (11),

End for

Find the fitness value of Best hunt agent (Y_α), Second best hunt agent (Y_β), Third best (Y_δ) hunt agents using equation (12),

renew the esteem of Y_α , Y_β , Y_δ

Update the vectors a , A and C

$\hat{C} = \hat{C} + 1$ until $\hat{C} \geq \hat{C}_{\max}$

End

output Y_α

Algorithm 2: Pseudo code of AGWO

Step 1: Initialize the AGWO parameters are search agents (Y_s), vectors $\vec{a}, \vec{A}, \vec{C}$ and the most extreme number of cycles ($\hat{C}_{\max}$).

$$\vec{A} = 2\vec{a} \cdot r_1 - \vec{a} \tag{4}$$

$$\vec{C} = 2 \cdot r_2 \tag{5}$$

The estimations $\vec{a}$ directly diminish from 2 to 0 through the span of iterations and r_1, r_2 are random vectors in [0, 1]. The parameter $\vec{a}$ is linearly refreshed in each cycle to go from [2–0] as shown by the condition (6),

$$\vec{a} = 2 - t \cdot \frac{2}{\hat{C}_{\max}} \tag{6}$$

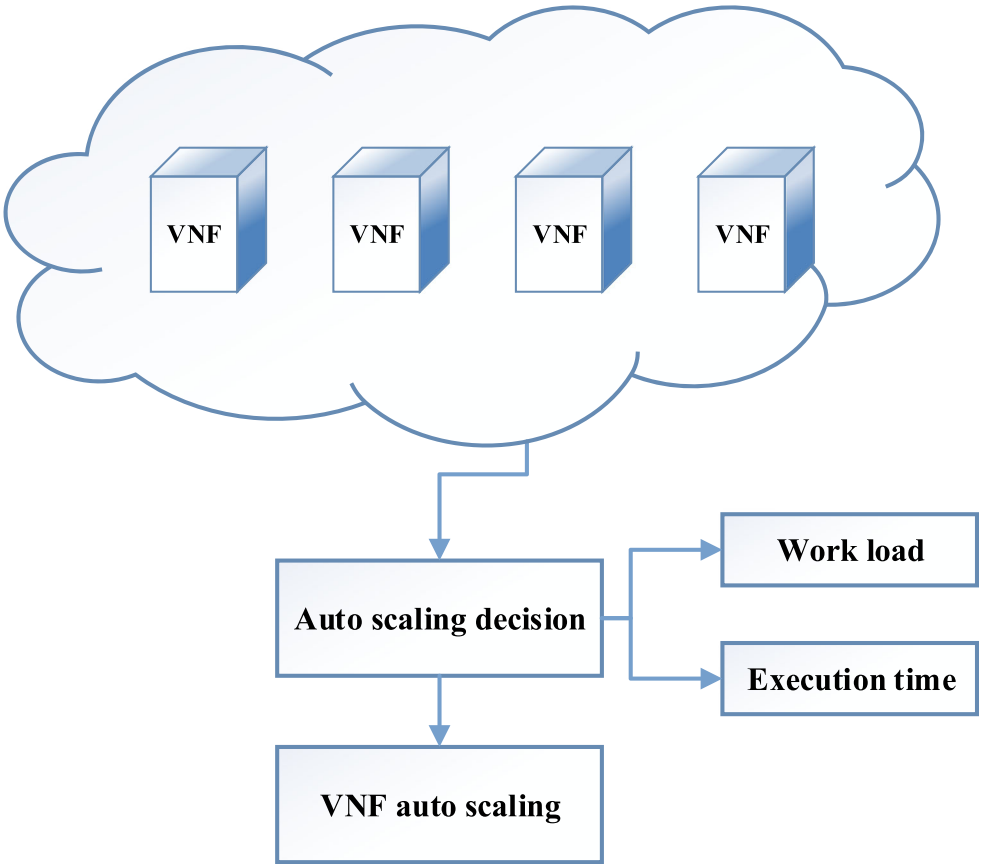

Fig. 2 Fuzzy logic based VNF auto-scaling schematic diagram

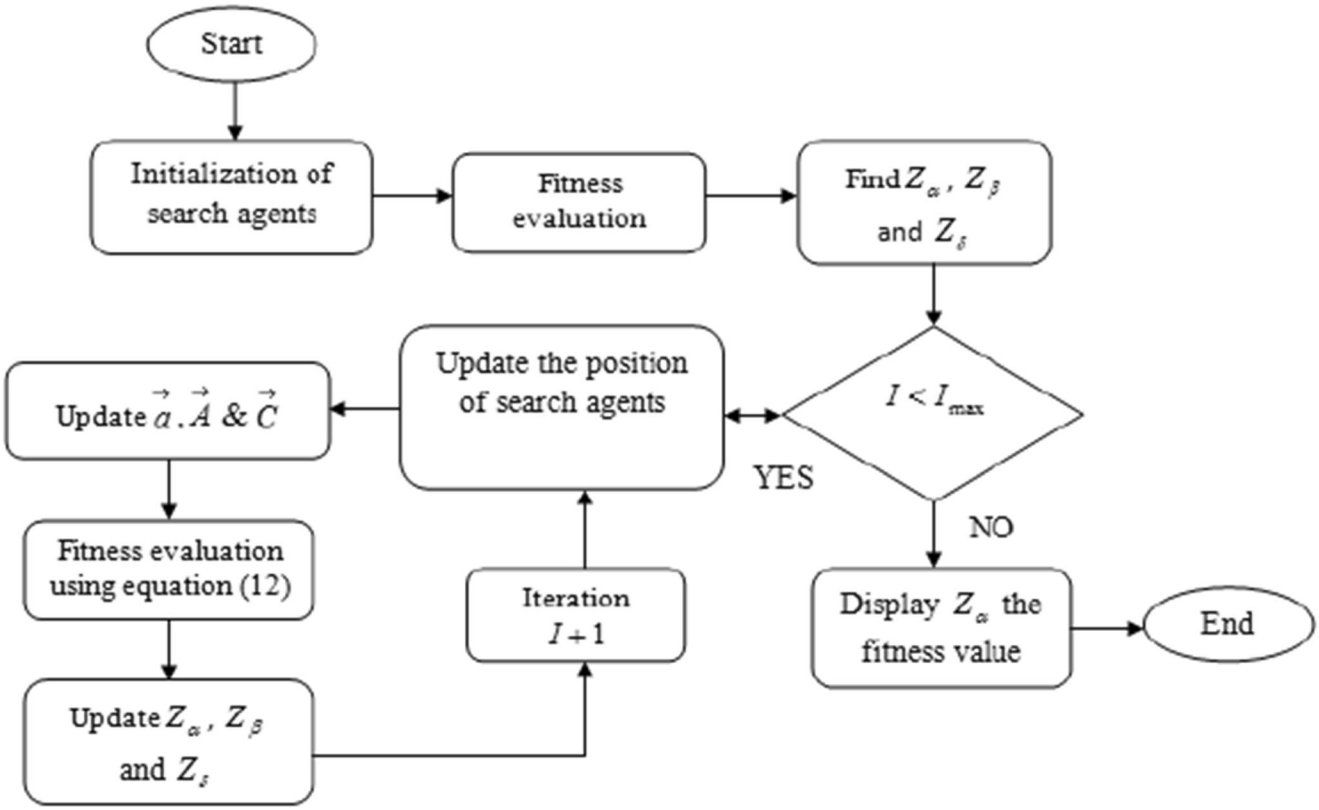

Fig. 3 Flow diagram of AGWO

Where,t is the iteration number and$\widehat{C}_{\max}$ is the total number of iteration took into consideration the optimization.

Step 2: Generate wolves haphazardly considering the size of the pack.
Step 3: Assess the fitness esteem regard of each hunt agent utilizing condition (7),

$$\overrightarrow{Y}(t+1) = \overrightarrow{Y}_p(t) + \overrightarrow{A}\cdot\overrightarrow{D} \tag{7}$$

Where, $\overrightarrow{D}$ is portrayed in condition (8) and$t$ is the iteration number, $\overrightarrow{A}$, $\overrightarrow{C}$ are coefficient vectors, $\overrightarrow{Y}_p$ is the prey position, and $\overrightarrow{Y}$ is the grey wolf position.

$$\overrightarrow{D} = \left|\overrightarrow{C}\cdot\overrightarrow{Y}_p(t) - \overrightarrow{Y}(t)\right| \tag{8}$$

Step 4: Find the most excellent hunt agent (Y_α), the second most excellent hunt agent (Y_β), and the third most excellent hunt agent (Y_δ) using condition (9),

$$\overrightarrow{Y}_1 = \overrightarrow{Y_\alpha} - \overrightarrow{A_1}.\left(\overrightarrow{D_\alpha}\right), \overrightarrow{Y}_2 = \overrightarrow{Y_\beta} - \overrightarrow{A_2}.\left(\overrightarrow{D_\beta}\right) \text{ and } \overrightarrow{Y}_3 = \overrightarrow{Y_\delta} - \overrightarrow{A_3}.\left(\overrightarrow{D_\delta}\right) \tag{9}$$

$$\text{Where, } \overrightarrow{D}_\alpha = \left|\overrightarrow{C_1}\cdot\overrightarrow{Y}_\alpha - \overrightarrow{Y}\right|, \overrightarrow{D}_\beta = \left|\overrightarrow{C_2}\cdot\overrightarrow{Y}_\beta - \overrightarrow{Y}\right| \text{ and } \overrightarrow{D}_\delta = \left|\overrightarrow{C_3}\cdot\overrightarrow{Y}_\delta - \overrightarrow{Y}\right| \tag{10}$$

Step 5: Update the location of the existing hunt agent utilizing condition (11),

$$\overrightarrow{Y}(t+1) = \frac{\left(\overrightarrow{Y_1} + \overrightarrow{Y_2} + \overrightarrow{Y_3}\right)}{3} \tag{11}$$

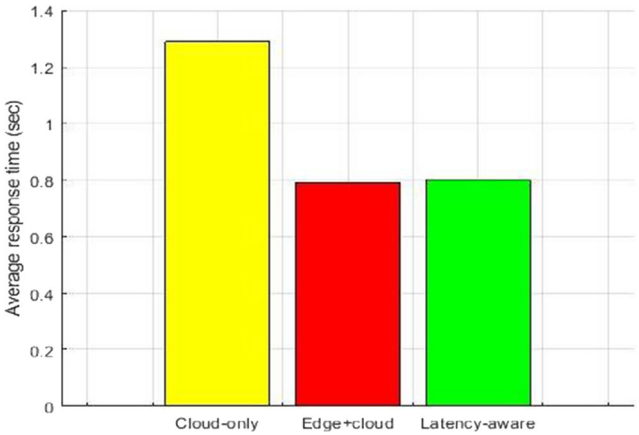

Fig. 4 Average response time for all applications

Step 6: Evaluate the fitness value for all hunts. Execution time and Workload can be limited just when the efficient group of tasks is fed to Virtual Machines (VMs). The fitness assessment function is modified as,

$$\widetilde{F} = \lambda e^{-E_T} + \gamma e^{-W_l\left(VNF_{cpu}\right)} + \overrightarrow{Y}(t+1) \tag{12}$$

Where $E_T$represents the execution time and $W_l(VNF_{cpu})$represents workload λ represents the weight factor of execution time and γ denotes the weight factor of the workload. This fitness assessment limits the two major resource provisioning objectives are execution time and workload.

Step 7: Update the estimation of $\overrightarrow{Y_\alpha}$, $\overrightarrow{Y_\beta}$ and $\overrightarrow{Y_\delta}$.
Step 8: Check for halting condition that is, regardless of whether the iteration reaches the most extreme, then yield the best estimation of solution else goes to step 5. The flow diagram of the AGWO algorithm is shown in Fig. 3.

The presented technique utilizes both edge and cloud resources to provision VNFs. In the case of VNF overloading in edge nodes, the algorithm relocates VNFs to the central cloud to use more resources. But the existing works lack the consideration in the application's latency necessities which can restrict the migration to the cloud. The presented methodology considers the latency requirements of applications, whereas the other workload of applications to be redirected to the cloud to utilize its adequate resources.

4 Results and discussion

The implementation of our presented effective latency aware resource provisioning is performed in the working platform of MATLAB 2018a. To research, the performance of the

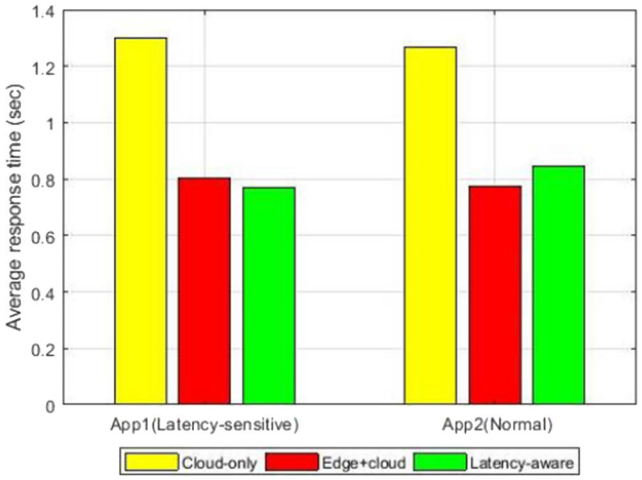

Fig. 5 Average response time of each application

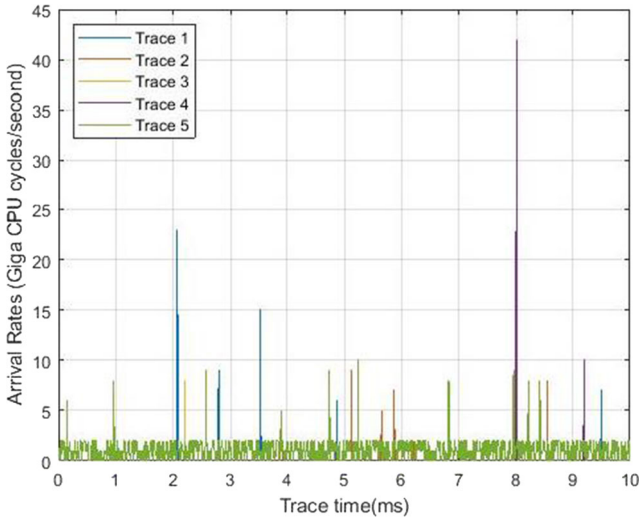

Fig. 6 Arrival rates of mobile requests

proposed work different of performance estimates such as system cost, arrival rate, average response time with existing resource provisioning schemes like Optimal Resource Provisioning with Hybrid Strategy (ORP-HS) [17], Optimal Resource Provisioning with On-Demand instances (ORP-OD) [17], Local first algorithm, Cloud-first algorithm and Optimal Resource Provisioning with Reserved instances (ORP-R) [17] are examined.

4.1 Dataset description

This paper works Google cluster usage traces dataset [28] to assess the presented system. In a Google cluster, a Google compute cell normally contains a set of machines that are associated with a high-bandwidth network. Subsequently, the mobile edge can be measured as a cloud of small range, user data of a Google compute cell is utilized to estimate the user data of the mobile edge. The Google cluster tracelogs4 record the information of tasks in a Google compute cell. The tasks are described by task event tables and task resource usage tables. Task event tables record task event information, such as event types (submit, schedule, fail, finish, etc.), job IDs, task indexes, and timestamps when these events happen.

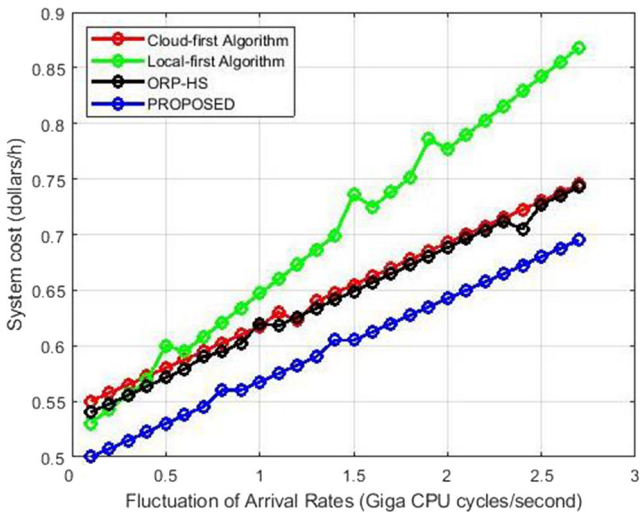

Fig. 7 System cost over the fluctuation of arrival rates

4.2 Average response time

It characterizes that the complete time taken to respond during the chosen time frame divided by the number of responses in the chosen period.

$$A_{response} = \frac{T}{R_n} \tag{13}$$

Here, $A_{response}$ signifies the Average response time, T signifies the total time taken to respond during the chosen time frame, and R_n indicates the number of responses in the chosen timespan. The response time of each request in the workload is assessed. The average response time of all workloads irrespective of the application with various VNF provisioning techniques is delineated in Fig. 4.

In Fig. 4, the existing Cloud-only resource provisioning demonstrates the outcome with just cloud assets exploited without utilizing any edge assets. VNFs are made and provisioned distinctly in the cloud assets; hence the average response time is altogether expanded because of the additional delay that all the packets must be transmitted to the central cloud. The following two outcomes are with exploiting edge resources alongside the cloud resources with the baseline algorithm (Edge + Cloud) and the presented effective latency aware provisioning. The average response time of the proposed strategy is diminished than the existing cloud-only and Edge + Cloud cases. The comparison analysis of the Average response time of every application is depicted in Fig. 5.

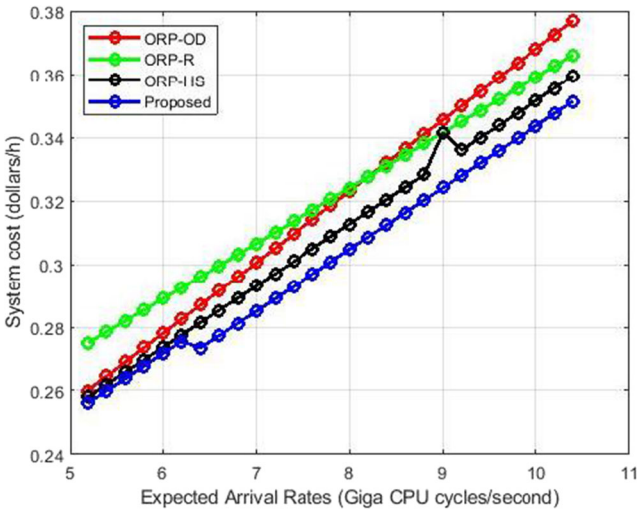

Fig. 8 Comparison of proposed with existing ORP methods for expected arrival rates

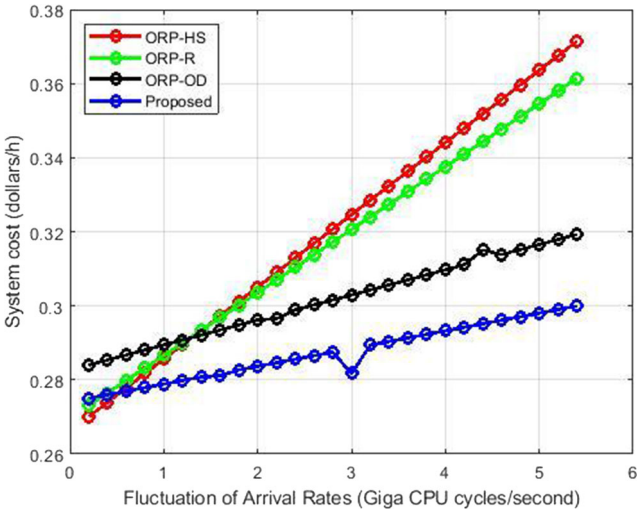

Fig. 9 Comparison of proposed with existing ORP methods for fluctuation of arrival rates

At the point when we measure the average response time for every application independently, the average delay is separated among applications and it is portrayed in Fig. 5. For the latency-sensitive application (App1), the average response time is decreased compared with the current time-critical applications.

4.3 Arrival rate

The arrival rate is the number of arrivals per unit of time. The arrival rate of computation requests can be determined as,

$$\overline{\lambda} = R^{comp} \cdot A^{task} \tag{14}$$

Where, A^{task} represents the arrival rate of the tasks. The computation requests of the tasks can be calculated as,

$$R_{comp} = \left(t^{finish} - t^{schedule}\right) \cdot U^{cpu} \cdot C^{cpu} \tag{15}$$

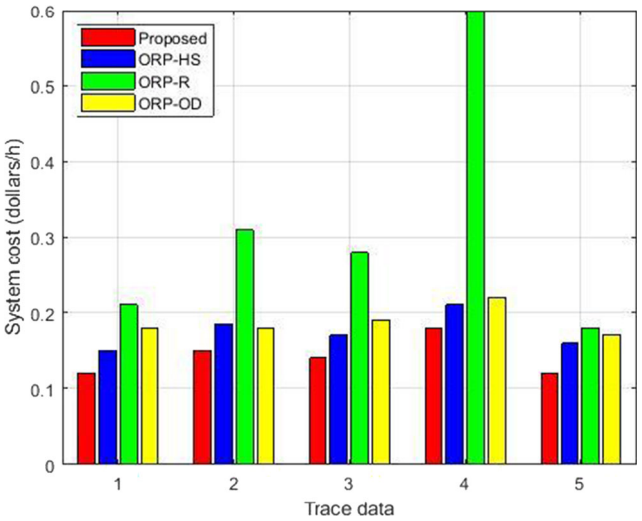

Fig. 10 Comparison of the system cost for varying traces

Here, t^{finish} and $t^{schedule}$represent the timestamps the task is done and scheduled to machines individually,C^{cpu} indicates the average computation capacity of CPU in the Google.

cloud. The arrival rates of mobile requests for fluctuating traces are given in Fig. 6.

In mobile edge computing, computation requests are substantially more delay-sensitive and have fewer computation necessities than those in conventional cloud computing. Along these lines, the arrival rates of computation requests at the mobile edge can be acquired by a little altering the results of Google cluster trace logs, as appeared in Fig. 6. Five groups of trace results are represented in Fig. 6. It tends to be seen that Trace-data 2, 3, 4 significantly fluctuate while Trace-data 1 and 5 are less changed.

4.4 System cost

The presented latency aware resource provisioning can generally outperform the existing Optimal Resource Provisioning with Hybrid Strategy (ORP-HS), Optimal Resource Provisioning with On-Demand instances (ORP-OD), and Optimal Resource Provisioning with Reserved instances (ORP-R) in diminishing framework cost and managing elements of mobile requests. Along these lines, in the resource provisioning for delay-sensitive dominant mobile requests, the exhibition of the different strategies are assessed and the examination results appear in Fig. 7.

In ORP-HS, the framework cost can be limited by accomplishing an optimal balance among lower pricing rates and higher asset usage. In this way, the local-first and ORP-R strategies can't scale well with the dynamics of mobile requests, yet the ORPOD and ORP-HS strategies can accomplish higher adaptability by flexible on-demand instances. The comparison investigation of proposed with existing ORP schemes for expected arrival rates is depicted in Fig. 8.

The results of the proposed ORP methods are looked at, as appeared in Fig. 8. As the expected arrival rates upsurge, mobile requests become less varied. By the by, the pricing rate of the reserved instances is greatly lower than on-demand instances. Thus, the system cost of ORP-OD rises quicker than the ORP-R strategy. In the proposed ORPHS scheme, the system cost can be essentially decreased by exploiting the advantages of both on-demand instances and reserved instances. The comparison analysis of proposed with existing ORP schemes for fluctuation of arrival rates is depicted in Fig. 9.

In ORP-HS, the system cost can be limited by accomplishing an optimal balance between a lower pricing rate and higher resource use. The comparison results of system cost for differing traces comparison among three ORP schemes are depicted in Fig. 10.

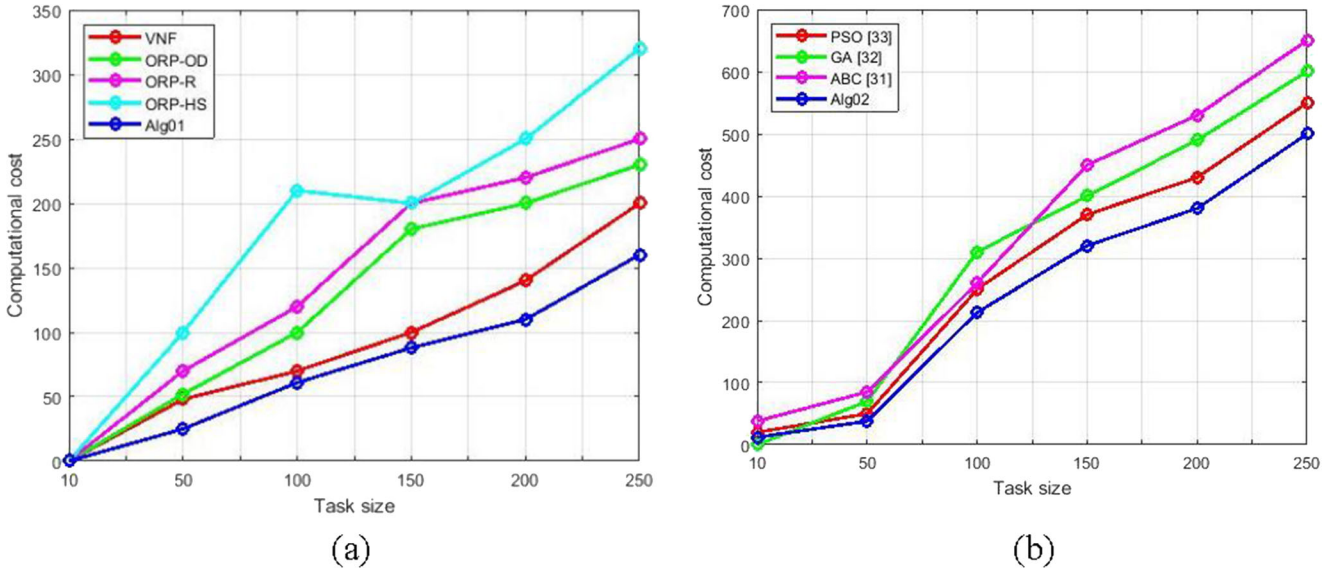

Fig. 11 Performance of Alg01 ORP-R, ORP-HS, ORP-OD, and VNF, by varying the task Size. **a** Computational cost **b** Running time

The outcomes that appeared in Fig. 10 portray that the proposed scheme considerably results in a lesser cost than other ORP-R, ORP-HS, ORP-OD schemes. The existing resource provisioning procedures results in Low resource utilization and high framework cost are acquired when dealing with dynamic requests. The proposed resource provisioning results the high resource usage and low system cost.

4.5 Performance evaluation of algorithms

4.5.1 Algorithm −1 evaluation

We first investigate the performance of Algorithm 1 against that of three baseline heuristics ORP-R, ORP-HS, ORP-OD, and VNF [21] for the auto-scaling problem of a VFV-enabled request admission, by varying the task size from 10 to 250.

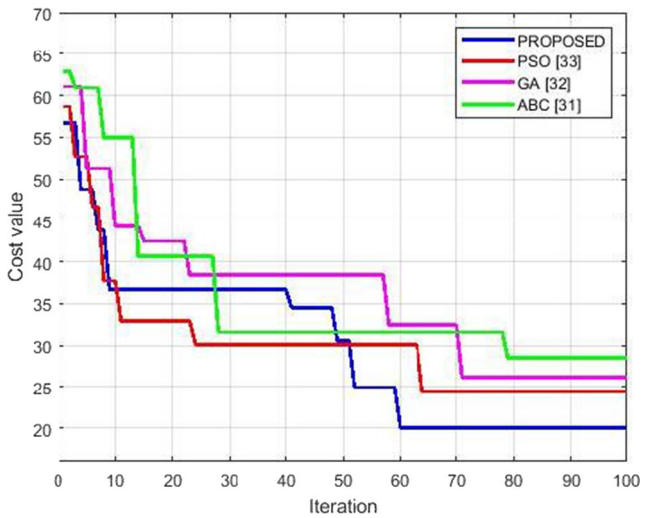

Fig. 12 Convergence graph of proposed and existing optimization techniques

Figure 4 illustrates the Computational cost and running time of the four mentioned algorithms. From Fig. 11 (a), we can see that Algorithm 1 achieves a much lower computational cost than those four Existing algorithms. The reason behind is that Algorithm 1 jointly considers the placement of VNF instances and data traffic routing for a request admission, it also makes a smart decision between using an existing VNF instance or creating a new Fuzzy logic based VNF instance. Figure 11 (b) plots the running time curves of the four Existing algorithms.

4.5.2 Algorithm −2 evaluation

In the proposed system, the convergence occurs between fitness and number of iterations using proposed enthalpy based grey wolf optimization is given in Fig. 12 and the convergence occurs in existing ABC [31], GA [32], and PSO [33] techniques are given in Fig. 11 respectively.

In Fig. 12, the proposed AGWO convergence is occurring in iteration number 60, and existing PSO, GA, and ABC techniques convergence is occurring in iteration number 64, 71, 79 respectively.

We then study the performance of Algorithm 2 against a heuristic PSO, GA, and ABC for the throughput maximization problem, by varying the task size from 10 to 250 for a set of multicast requests. Figure 13 plots the performance curves of the four algorithms. It can be seen from Fig. 13 (a) that Algorithm 2 outperforms the benchmark PSO, GA, and ABC in all cases, and their performance gap becomes larger and larger with the increase in task size. Specifically, the network throughout achieved by Algorithm 2 is 15.7% and 20.5%, 25.6%, 30.6% higher than that by algorithm PSO, GA, and ABC. Figure 13 (a) Computational cost by Algorithm 2 is 10.4% and 22.3%, 24.4%, 21.1% higher than

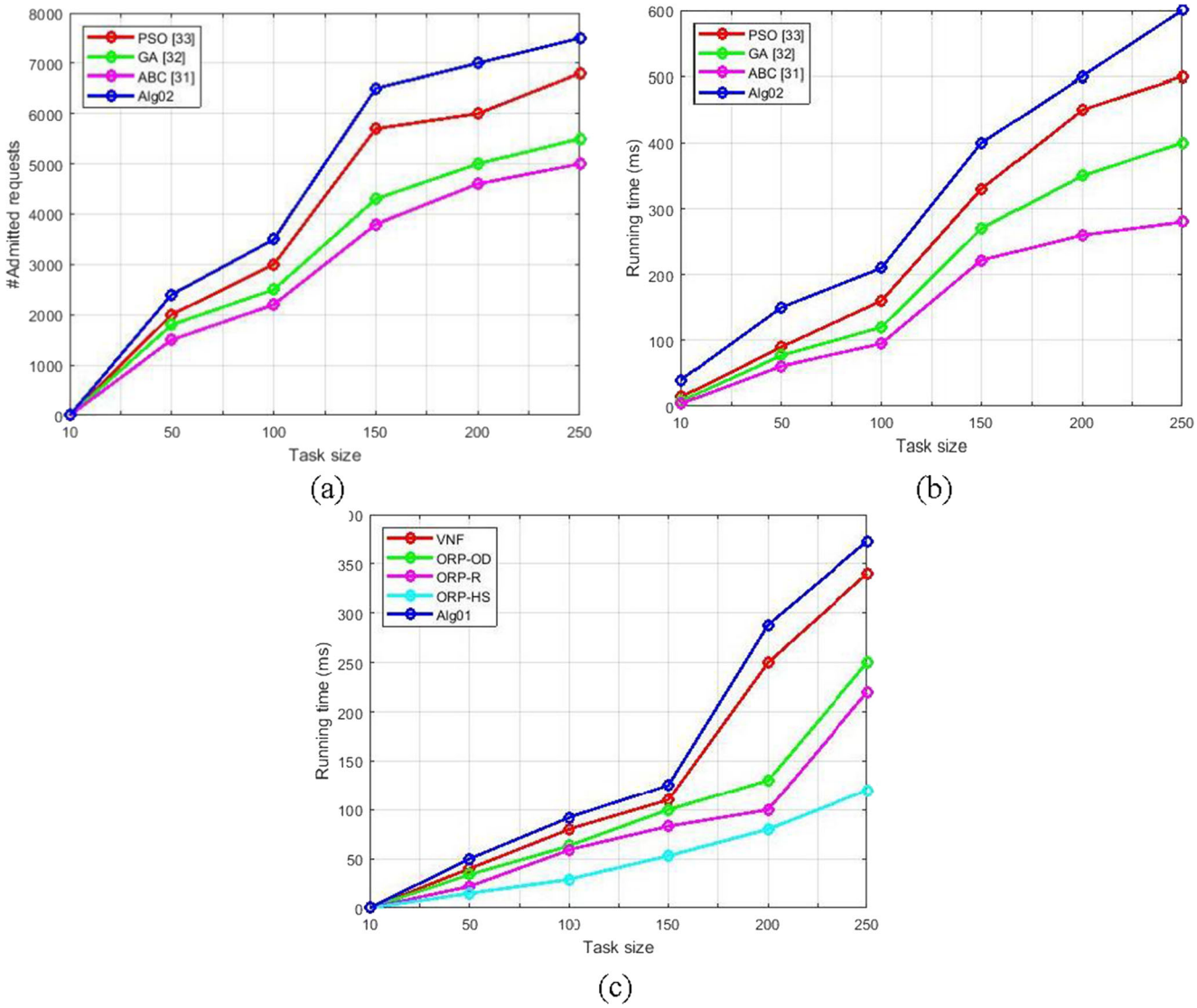

Fig. 13 Performance of Alg02, PSO, GA and ABC, and VNF, by varying the task size. **a** Throughout **b** Computational cost **c** Running time

that by PSO, GA, and ABC, when the task size is set at 50 and 250, respectively. Figure 5 (c) depicts the running times of the mentioned two algorithms. It can be seen that Algorithm 2 takes a longer time than that of algorithm PSO, GA, and ABC for finding a more accurate solution.

The failure rate, Memory utilization, and CPU utilization of the proposed AGWO work is compared with the existing PSO, GA, ABC is explained in Tables 1 and 2. Here the proposed work provides the failure rate by utilizing the Google cluster usage traces dataset 250 and 512 tasks are 1.6620, and 0.3216. Then the Memory utilization of the proposed work by utilizing the Google cluster usage traces dataset 250 and 512 tasks are 1.5700 and 1.1380. Finally, the CPU utilization of the proposed work by utilizing the Google cluster usage traces dataset 250 and 512 tasks are 1.4868 and 1.3315and the same is calculated in the existing algorithm.

Table 1 comparison of the proposed work with the existing in terms of failure rate, Memory utilization, and CPU utilization for 250 tasks

Algorithm	Task	Failure rate	Memory utilization	CPU utilization
Proposed AGWO	250	1.6620	1.5700	1.4868
PSO [33]		3.3344	4.4566	1.1652
GA [32]		2.7604	8.6765	2.4909
ABC [31]		3.8704	9. 7065	3. 9009

Table 2 comparison of the proposed work with the existing in terms of failure rate, Memory utilization, and CPU utilization for 512 task

Algorithm	Task	Failure rate	Memory utilization	CPU utilization
Proposed AGWO	512	0.3216	1.1380	1.3315
PSO [33]		1.7777	0.6184	2.3131
GA [32]		1.7343	9.1742	2.5806
ABC [31]		1.4443	10.1332	2.2843

5 Conclusion

This paper exhibited proficient latency-aware resource provisioning in a cloud-assisted mobile edge framework. The presented effective resource provisioning is achieved by fuzzy logic based auto-scaling for the overloaded VNFs that require more resources because of the progressively expanded amount of the network packets. Thusly, the SFC requests are scheduled to the cloud-assisted edge network viably utilizing AGWO based resource provisioning. The performance of the proposed methodology is analyzed with existing resource provisioning schemes, for example, ORP-HS, ORP-OD, Local first algorithm, Cloud-first algorithm, and ORP-R in regard to system cost over fluctuation, arrival rate, and average response time. The exploratory outcomes exhibit that the presented resource provisioning is better than the current strategies.

References

1. Gubbi J, Buyya R, Marusic S, Palaniswami M (2013) Internet of things (IoT): a vision, architectural elements, and future directions. Futur Gener Comput Syst 29(7):1645–1660
2. Yadav R, Zhang W, Kaiwartya O, Singh PR, Elgendy IA, Tian Y-C (2018) Adaptive energy-aware algorithms for minimizing energy consumption and SLA violation in cloud computing. IEEE Access 6:55923–55936
3. Savaglio C, Fortino G, Zhou M (2016) Towards interoperable, cognitive and autonomic IoT systems: An agent-based approach. In 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT): 58–63, IEEE
4. Fernando N, Loke SW, Rahayu W (2013) Mobile cloud computing: a survey. Futur Gener Comput Syst 29(1):84–106
5. Elgendy IA, El-kawkagy M, Keshk A (2015) An efficient framework to improve the performance of mobile applications. International Journal of Digital Content Technology and its Applications (JDCTA) 9(5):43–54
6. Elgendy MA, Shawish A, Moussa MI (2014) MCACC: New approach for augmenting the computing capabilities of mobile devices with Cloud Computing. In 2014 Science and Information Conference:79–86. IEEE
7. Hu YC, Patel M, Sabella D, Sprecher N, Young V (2015) Mobile edge computing—A key technology towards 5G. ETSI white paper 11(11):1–16
8. Zhang S, He P, Suto K, Yang P, Zhao L, Shen X (2017) Cooperative edge caching in user-centric clustered mobile networks. IEEE Transactions on Mobile Computing 17(8):1791–1805
9. Sarkar S, Chatterjee S, Misra S (2015) Assessment of the suitability of fog computing in the context of internet of things. IEEE Transactions on Cloud Computing 6(1):46–59
10. Aazam M, Huh E-N (2015) Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT. In 2015 IEEE 29th International Conference on Advanced Information Networking and Applications:687–694, IEEE
11. Jia M, Cao J, Liang W (2015) Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks. IEEE Transactions on Cloud Computing 5(4):725–737
12. Yang P, Zhang N, Zhang S, Yu L, Zhang J, Shen XS (2018) Content popularity prediction towards location-aware mobile edge caching. IEEE Transactions on Multimedia 21(4):915–929
13. Chen TY-H, Ravindranath L, Deng S, Bahl P, Balakrishnan H (2015) Glimpse: continuous, real-time object recognition on mobile devices. In Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems:155–168
14. Ha K, Pillai P, Richter W, Abe Y, Satyanarayanan M (2013) Just-in-time provisioning for cyber foraging. In Proceeding of the 11th annual international conference on Mobile systems, applications, and services:153–166
15. Liang B (2017) Mobile edge computing. In: Wong VWS, Schober R, Ng DWK, Wang L-C (eds) Key technologies for 5G wireless systems. University Press, Cambridge
16. Ma X, Zhang S, Li W, Zhang P, Lin C, Shen X (2017) Cost-efficient workload scheduling in cloud assisted mobile edge computing. In 2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS):1–10, IEEE
17. Ma X, Zhang S, Yang P, Zhang N, Lin C, Shen X (2017) Cost-efficient resource provisioning in cloud assisted mobile edge computing. In GLOBECOM 2017–2017 IEEE Global Communications Conference:1–6, IEEE
18. Avasalcai C, Dustdar S (2019) Latency-aware distributed resource provisioning for deploying iot applications at the edge of the network. In Future of Information and Communication Conference: 377–391, Springer, Cham
19. Guo J, Li C, Yi C, Luo Y (2019) On-demand resource provision based on load estimation and service expenditure in edge cloud environment. J Netw Comput Appl 102506
20. Elgendy IA, Zhang W, Tian Y-C, Li K (2019) Resource allocation and computation offloading with data security for mobile edge computing. Futur Gener Comput Syst 100:531–541
21. Son J, Buyya R (2019) Latency-aware virtualized network function provisioning for distributed edge clouds. J Syst Softw 152:24–31
22. Li C, Sun H, Tang H, Luo Y (2019) Adaptive resource allocation based on the billing granularity in edge-cloud architecture. Comput Commun 145:29–42
23. Chen X, Li W, Lu S, Zhou Z, Xiaoming F (2018) Efficient resource allocation for on-demand mobile-edge cloud computing. IEEE Trans Veh Technol 67(9):8769–8780
24. Saremi S, Mirjalili SZ, Mirjalili SM (2015) Evolutionary population dynamics and grey wolf optimizer. Neural Comput & Applic 26(5):1257–1263

25. Fan Q, Ansari N (2019) On cost aware cloudlet placement for mobile edge computing. IEEE/CAA Journal of Automatica Sinica 6(4):926–937
26. Zhang PY, Shu S, Zhou MC (2018) An online fault detection model and strategies based on SVM-grid in clouds. IEEE/CAA Journal of Automatica Sinica 5(2):445–456
27. Huang J, Li S, Duan Q (2017) Constructing multicast routing tree for inter-cloud data transmission: an approximation algorithmic perspective. IEEE/CAA Journal of Automatica Sinica 5(2):514–522
28. http://code.google.com/p/googleclusterdata/wiki/ClusterData2011_1
29. Zhang Y, Zhou P, Cui G (2018) Multi-model based PSO method for burden distribution matrix optimization with expected burden distribution output behaviors. IEEE/CAA Journal of Automatica Sinica 6(6):1506–1512
30. Gao S, Zhou MC, Wang Y, Cheng J, Yachi H, Wang J (2018) Dendritic neuron model with effective learning algorithms for classification, approximation, and prediction. IEEE transactions on neural networks and learning systems 30(2):601–614
31. Van Do T, Do NH, Kispal I, Galambosi N, Rotter C, Nemeth L (2018) A big switch abstraction to support service function chaining in cloud infrastructure. In 2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN):1–5. IEEE
32. Carpio F, Dhahri S, Jukan A (2017) VNF placement with replication for Loac balancing in NFV networks. In 2017 IEEE International Conference on Communications (ICC):1–6. IEEE
33. Qi D, Shen S, Wang G (2019) Virtualized network function consolidation based on multiple status characteristics. IEEE Access 7: 59665–59679

Rajasekhar Bandapalle Mulinti received his M.Phil. In Computer Science from Sri Krishnadevaraya University, Anantapur, India, in 2011. He received M.Sc. in Computer Science from S.K. University, Anantapur, India, in 2005. He is currently pursuing his Ph.D. in Computer Science and Technology at Sri Krishnadevaraya University, Anantapur. A.P. India. His current research Interest includes Computer Networks and Network Security.

M. Nagendra , Professor in the Department of Computer Science and Technology, Sri Krishnadevaraya University, Ananthapuaramu, A.P. India.