Received: August 25, 2022. Revised: January 10, 2023.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

LotkaVoltera and Elman Hebbian Recurrent Neural Network Cache-Based Resource Allocation in Fog Environment

S.V. Nethaji[1]* **M. Chidambaram[1]**

[1]*PG & Research Department of Computer Science, Rajah Serfoji Government College(Autonomous), (Affiliated to Bharathidasan University, Tiruchirappalli),Thanjavur, India.*
* Corresponding author's Email: nethajisv@gmail.com

Abstract: The internet of things (IoT) affects everyday life because digital technology improves. IoT is a group of devices with sensors that talk to each other to reach a goal. IoT systems have traditionally been built on top of cloud computing (CC). IoT devices are slow because CC data centres are separated from them. This slows down the speed at which real-time applications respond. In addition, IoT devices send a lot of data to the cloud to be processed, which overloads the cloud. Edge computing can stop IoT devices from being slow or overloaded. Fog computing (FC) is a way to get services at the edge of a network. With locationawareness, the FC cuts down on latency and overloading. Bandwidth and jitter must be looked at during the process of allocating resources. In this work, the Lotka-Voltera load balancer and elman hebbian-recurrent neural network cache (LV-EHRCC) are proposed for allocating resources in an FC context. LV-EHRCC is made up of load balancing and allocating resources. First, the LotkaVoltera traffic load balancer model is used to increase the amount of Bandwidth available for load balancing. Second, an Elman Hebbian-Recurrent Neural Network model for allocating cached resources efficiently is made for the best load-balanced FC context. Simulations test what will happen. The load balancing capacity of the proposed scheme is 93.25 and attains the highest Bandwidth of 45. In FC simulations, the LV-EHRCC method improves the efficiency of load balancing in terms of Bandwidth, makespan, and jitter rate. The simulation results back up our study and show that LV-EHRCC is better than the benchmark approaches when they are compared.

Keywords: IoT, Cloud computing, Fog computing, LotkaVoltera, Load balancer, Virtual machine, Elman, Hebbian.

1. Introduction

IoT has recently given rise to a fresh wave of embedded internet-connected applications. Numerous different applications can be deployed with cloud computing (CC), and its concentric administration provides effective object communication. Scalable systems can be managed by IoT thanks to their enormous data storage, computing power, and resource provisioning. IoT's low latency is used in CC's centralized framework despite its ability to handle scalability issues. FC can address these issues. Fog effective prediction and resource allocation methodology (EPRAM) was presented in [1]. The prediction method assisted EPRAM in resource management. EPRAM consisted of the following modules: Data pre-processing module (DPM), resource allocation module (RAM), and effective prediction module (EPM). To forecast the target field using one or more predictors, EPM employed PNN. Here, utilizing user IoT data, the PNN predicts the likelihood of a heart attack and takes appropriate action. The objective was to improve QoS metrics, including response time, Bandwidth, and energy consumption while lowering latency. These elements were found with deep reinforcement learning (RL). Although response time, Bandwidth, and energy usage all improved, load balancing effectiveness was not given priority. To ensure processing and IoT data transfer to the CC environment, [2] suggests load balancing for IoT gateways. This topology suspended the FoT data streams to provide scalability and latency. Cloud and fog data traffic saw the least latency and capacity thanks to the effective management of the software defined network (SDN) controller. We measured response time, active time, and missing samples. The

results defeated round robin and least connection. IoT with FC involves hardware and software resource allocation; response time and busy time were enhanced; bandwidth and jitter rate was not. The neural network and meta-scheduler in [3] effectively predicted fog resources. Five algorithms were employed to identify the best help. [4] examines issues with the fog computing (FC) environment. Offloading energy and data from consumer devices with limited resources to cloud infrastructure led to system optimization and better performance. Fog unloading was thoroughly investigated in [5]. One of the most significant advancements in recent years has been moving data control and storage to the cloud to satisfy traffic demands. It affects latency by causing network delays. Fog computing, which moves control and data storage to the network edge, has become unavoidable in recent years as a solution to this problem. [6] offered a method for allocating resources and offloading work in multi-fog node systems. The average success rate and partial observability were enhanced using deep Q network (DQN) and deep convolutional Q network (DCQN). lack of load balance. [7] recommended a load-balancing method using cat swarm optimization to increase throughput and lower energy usage. In this study, a load-balancing and resource-provisioning strategy are proposed and implemented. The suggested approach combines optimization and deep neural networks to increase Bandwidth, jitter, makespan, and load balancing. The FC environment's best load balancing and resource distribution are made possible by the elman hebbian-recurrent neural network cache resource allocation (LV-EHRCC) and LotkaVoltera load balancer.

1.1 Contributions

Work contributions include:

• Propose LotkaVoltera load balancer and elman hebbian-recurrent neural network cache resource allocation (LV-EHRCC) in fog computing environment for significant prediction by combining a novel load balancing and resource allocation model, ensuring accurate balancing analysis with minimum Bandwidth and jitter.
• A LotkaVoltera traffic load balancer model to minimize Bandwidth and improve load balancing during user request jobs.
• Elman hebbian-recurrent neural network-based cache resource allocation model enables resilient and accurate decision-making based on studied load and optimal resource allocation across user-requested tasks.
• Extensive experimental assessment of LV-EHRCC approach against EPRAM and load balancing for FoT-Gateways to show the predictive analytical performance of the suggested method.

1.2 Organization of the paper

Here's how the rest of the paper is organized. A literature review of load balancing and resource allocation is presented in section 2. The proposed methodology LV-EHRCC, resource allocation in a fog computing environment is presented in section 3. The experimental setup is described in section 4, results and analysis are discussed in section 5, and conclusions are presented in section 6.

2. Related works

The upcoming fog computing and the Internet of Things IoT are said to be mandatory for next-generation communication applications. However, the communication capacity was found to be constrained upon comparison with the increase in internet of things (IoT) devices. Moreover, the optimal allocation of distinct tasks ignites the load-balancing aspect in the FC network to ensure optimal resource allocation.

In [8], a fog load balancing problem was designed,considering the communication and computation aspects to reduce the cost involved in load balancing. However, processing time, another factor involved during load balancing, should also be considered. Finally, in [9], home edge computing was designed based on clustering and load-balancing technique. With this,the congestion was avoided, but it also resulted in the minimization of latency.

Despite the outstanding work performed to enhance fog computing applications, task scheduling is still a significant concern. Owing to this, a novel multiobjective method was presented in [10] based on combining marine predator's algorithm with the polynomial mutation mechanism. With this, efficient task scheduling in FC environments was ensured. Furthermore, an outpouring in the significance of sensors and real-time monitoring factors has led to combining two powerful techniques: the cloud computing environment and the internet of things (IoT). Moreover, large-stream data processing has resulted in yet another novel approach, fog computing.

A survey of application algorithms for fog computing was investigated in [11]. In [12], there were five essential factors: concerns in fog computing, optimization process involved in fog and IoT, compared scheduling algorithms, rationalized the scheduling patterns and measures were also taken

for improving scheduling. A protection mechanism for edge computing was analyzed in [13].

Real-time decision-making is said to be arrived at when applied with latency-aware resource allocation. Fog nodes would ensure optimized decision-making when employed in a cloud computing environment. In [14], an efficient resource allocation and fault tolerance method for the fog layer was proposed. Also,the recovery time involved in failure was said to be improved. In [15] log likelihood ratio was measured for enhanced decision-making in a cloud environment.

To attain significant resource allocation and minimize users' computing time, the virtual machine (VM) allocation must be optimized. A new multiobjective optimization method with dynamic resource allocation combining the present state and future predicted data concerning each load, virtual machine relocation cost and new VM stability were also considered comprehensively. Also, a multiobjective optimization genetic algorithm was presented to address the issues concerning time and VM allocation [16]. Yet another method for data reduction was proposed in [17] by employing a naïve bayes classifier. Finally, in [18], by introducing mixed integer nonlinear optimization offloading was introduced to ensure smooth computing resource allocation.

The explosive evolution of small cell base stations (SBSs) delegated with computing potentialities presents one of the most ingenious factors applied as far as fifth-generation (5G) cellular networks are concerned to address data explosion and ultralow latency. In [19], green-based edge network management (GENM) algorithm was proposed that ensured green-based load balancing in base station BSs and reduced consumption of energy within the multi-access edge computing (MEC) server. In [20], another hybridization of the heuristics technique was applied to cope with the load balancing issue.

[21] models resource optimization in fog computing with shift-invariant deep convolutive load balancing technique in simulated fog computing environment using ifogsim.

In a fog computing environment, another [22] load balancing method using the differential evolution-based grey wolf optimization model and the resource allocation method utilizing the stochastic gradient and deep reinforcement learning-based resource allocation model are examined.

The existing approaches face the issues such as proficient allocation of resources and balancing the load. However, the current system does not accomplish accurate and effective decision-making without latency.

From the above research gaps, it has been analyzed that previous works on load balancing in the FC environment has given little significance to bandwidth-improved load balancing and jitter-optimized resource allocation. Therefore, present research on load balancing and resource provisioning has been done to understand better the issues considered earlier and to bridge the gaps in the previous studies.

3. Methodology

This section presents a three-plane LV-EHRCC resource allocation method of an FC network. The first tier of the LV-EHRCC process consists of sensors NEC personal cloud traces, for example, volume, node, session and request types and transmits to the fog nodes. The fog nodes constitute the second place of the method. The fog nodes analyze the data or data packets obtained from IoT devices, and the results are sent back for further analysis. To ensure that all the combinations of workloads and providers are addressed in a real-time environment, the fog nodes are placed at the edge of the network,i.e. adjacent to the IoT devices. The results of combinations of workloads and providers are also stored in the cloud data centre present at the cloud plane.

Moreover, the transmission link between a cloud server and fog nodes is constructed via a proxy server. The cloud data centre is predominantly utilized to ensure optimal data centres for storage. The fog computing-based method of our proposed method is shown in Fig. 1. It shows the three FC planes: end, fog, and cloud. End plane: contains end devices $D=D_1$, D_2..., Dm or IoT devices and is FG's data source. Let's assume Bandwidth and jitter categorize user-requested jobs. Then, for each user-requested task, Bandwidth and jitter are pre-arranged based on the number of instructions or data packets.

Fog plane: fog nodes $FN=FN_1,FN_2,...,F_N$ n next to end devices $D=D_1,D_2,...,D_m$. Data exchanges between fog nodes (FN) and end devices D expose delay. Fog nodes are co-located with base stations to which IoT devices are attached. Offloaded jobs require a central processing unit (CPU), memory, and storage in a virtual machine (VM) to process data packets for Bandwidth and jitter.

The confined resource volume at the fog node may not allow all tasks $T=T_1$, T_2..., Tm to be processed simultaneously due to the Bandwidth and jitter associated with offloading activities. So, tasks are queued up.

Cloud plane has an unlimited-resource cloud data centre (CDC). 'CDC' uses a single-user VM to

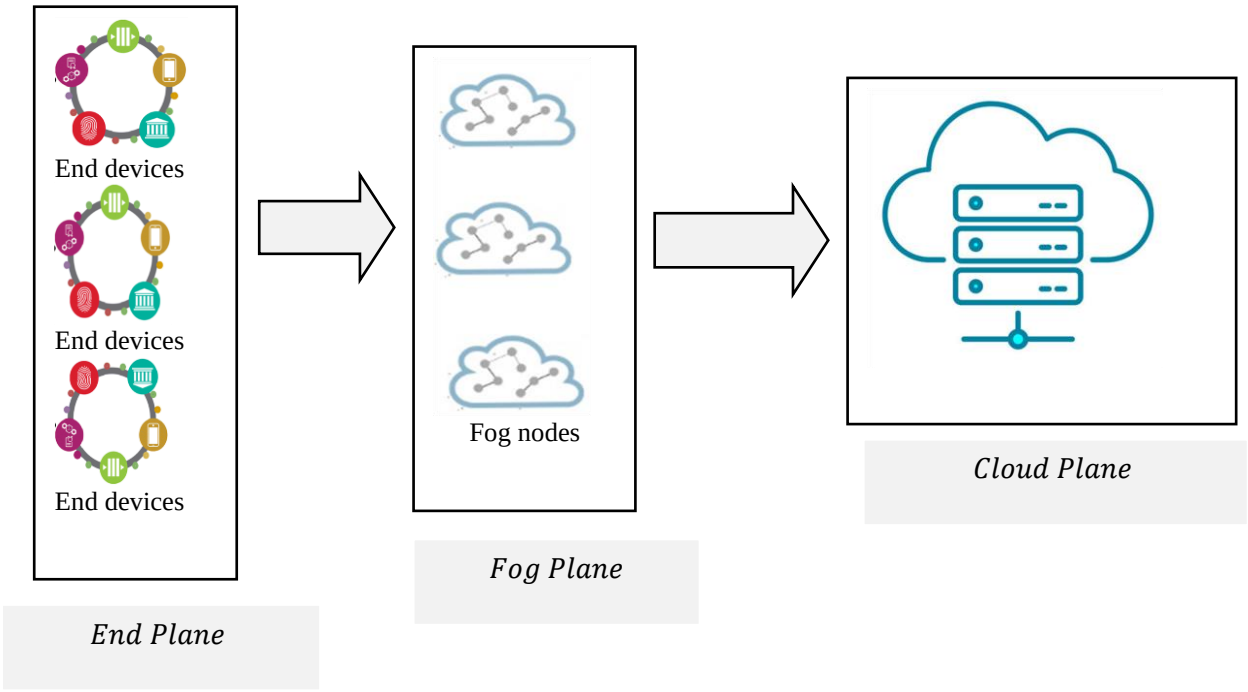

Figure. 1 Structure of three-plane optimal load balancing and resource allocation

process 'T' for best performance. The Bandwidth between FN and CDC and jitter causes data transmission latency. Based on the user-requested task processing arrangement in the three-plane fog computing network, processing (optimal load-balanced resource allocation) is split into two portions. User request task buffering (URTB) at fog nodes defines implementation sequences on user request task arrivals. LotkaVoltera traffic load balancer is used to accomplish URTB in this work. By calculating traffic load and looking at the queue using LotkaVoltera optimization, this approach achieves efficient load balancing with better Bandwidth, even with many user-requested jobs. Virtual machines are allocated according to user-requested tasks in a bandwidth-optimized load-balancing manner. Optimal resource allocation at fog nodes (ORA): The elman hebbian-recurrent neural network assigns user-requested tasks to fog resources in the waiting queue, so scheduled tasks can run.

With global cloud networks connecting millions of IoT devices to many servers, cloud user requests across devices take a long time. Due to this, cloud user requests on the edge of the cloud are moved to a costly fog layer. Therefore, optimized load balancing between fog and cloud layers is required for excellent service quality and efficiency. This study introduces a LotkaVoltera traffic load balancer model that optimizes bandwidth rate by considering traffic load and LotkaVoltera function. Fig. 2 depicts the LotkaVoltera traffic load balancer structure. The following LotkaVoltera

Traffic load balancer model indicates that correlations between transfer speed and load are recognized using two steps. First, the actual traffic load is evaluated using the IoT device's transmission power, channel gain, and size. Second, a LotkaVoltera optimization function considers incoming and queued workloads' propagation speed and saturation rate.

This balances the demand, increasing Bandwidth. Let's assume that ‘L' is a location identifying three types of volumes ‘Volume', and IP address of node ‘node IP' in area ‘A', and IoT device ‘D' put at ‘P' location has transmission power ‘TransP(L)', channel gain ‘CG(L)', and noise power ‘2'.

$$SNR(L) = \frac{TransP(L) \times CG(L)}{\sigma^2} \quad (1)$$

The channel gain 'CG' is measured as follows.

$$CG(L) = 10 \log 10 \left[\frac{\lambda}{(2\pi D)}\right] \quad (2)$$

The channel gain for a location in region ‘A' is based on wavelength ‘' with IoT device ‘D' and base stations ‘BS' differentiated via distance ‘D'. If an IoT device ‘D' is associated with ‘j-th' base station ‘BS_j' with bandwidth ‘BW_j', then the IoT device's magnitude ‘Mag j (L)' is mathematically defined as follows.

$$Mag_j(L) = BW_j X \log 10\,[1 + SNR(L)] \quad (3)$$

From Eq. (1), a poisson process describes random occurrences (i.e., user-requested activities).

Moreover, the IoT device 'D' at location 'L' includes some traffic load for the base station. 'BS_j'. Then, the traffic load is mathematically stated as given below.

$$TL_j(L) = \frac{DPFR(L)XDP(L)\times DC_j(L)}{Mag_j(L)} \quad (4)$$

From the above Eq. (4), the traffic load at location 'L' '$TL_j(L)$' is measured based on the data packet flow rate '$DPFR(L)$', number of data packets '$DP(L)$' and dual criterion. '$DC_j(L)$' if the device is associated with the respective base station. 'BS_j' or not. Also,to improve the volume or number of information (i.e., data packets) transmitted at a specific time instance in a load-balanced fashion, in our work, the LotkaVoltera function is employed. The LotkaVoltera function also referred to as the predator-prey equations (i.e., user-requested tasks-tasks in the queue) denotes the pair of nonlinear differential equations in which two species interact (i.e., incoming tasks and the tasks already in the queue), one as a predator (i.e., user-requested tasks) and the other as prey (i.e., tasks in the queue). This is mathematically formulated as given below.

$$\frac{dT}{dt} = \alpha\, T - \beta TQ \quad (5)$$

$$\frac{dQ}{dt} = \delta TQ - \gamma Q \quad (6)$$

From Eqs. (5) and (6), and refer to the propagation rate of user-requested tasks, saturation rate of user-requested jobs, saturation rate of charges in a queue, and propagation rate of the queue concerning tasks 'T' in queue 'Q' correspondingly. By using the propagation and saturation rate, load balancing between user-requested jobs is ensured, increasing the maximum quantity of data or data packets transmitted over the internet in a given time. Below is LotkaVoltera's pseudo code.

In the LotkaVoltera above traffic load balancer algorithm, the traffic load is measured using the poisson process to improve Bandwidth first. Second, with the observed traffic load, a LotkaVoltera optimization function assesses the tasks that may be handled, balancing the load optimally. Third, user-requested tasks are distributed evenly between virtual computers.These two functions increase Bandwidth and load ba lance.

Input: Dataset 'DS', end devices '$D = D_1, D_2, \dots, D_m$', fog nodes '$FN = FN_1, FN_2, \dots, FN_n$', data packet '$DP = DP_1, DP_2, \dots, DP_k$', Virtual Machine '$VM$', tasks '$T = T_1, T_2, \dots, T_l$'
Output:Bandwidth-improved optimal load balancing
1: **Initialize** 'm, n, l, k', cloud data centre 'CDC', Base Stations 'BS', queue 'Q' 2: **Initialize** CPU 'CPU', memory 'M' and storage 'S', transmission power '$TransP(L)$' 3: **Initialize** '$\alpha = 0.1$', '$\beta = 0.02$', '$\gamma = 0.4$' and '$\delta = 0.02$' 4: **Begin** 5: **For** each Dataset 'DS' with end devices 'D', fog nodes 'FN' and data packet 'DP' 6: Evaluate Signal to Noise Ratio as in (1) 7: Estimate the magnitude of the IoT device as in (3) 8: **For** each Base Stations 'BS' 9: Evaluate traffic load as in (4) 10: **End for** 11: **For** each tasks 'T' and tasks in queue 'Q' 12: Evaluate predator and prey as in equations (5) and (6) 13: **End for** 14: **End for** 15: **End**

Algorithm 1 LotkaVoltera traffic load balancer

3.1 Elman hebbian-recurrent neural network-based cache resource allocation

Upon load balancing, user-requested tasks must be allocated to fog resources in the waiting queue, so planned tasks can be done on time. Despite various studies ensuring effective resource allocation, significant congestion is reported to cause dissimilarity in data packet flow between two systems or user-requested tasks, resulting in delay. This part presents an elman hebbian-recurrent neural network-based cache resource allocation model. In this model, all user-requested jobs are expected to be stored in the local cache and can be obtained directly from base stations without downloading from the cloud data centre. In elman hebbian-recurrent neural network-based cache resource allocation, inputs and outputs are provided to learn the mapping between inputs (i.e., load-balanced user-requested tasks) and outputs (i.e., optimal resource allocation). The elman

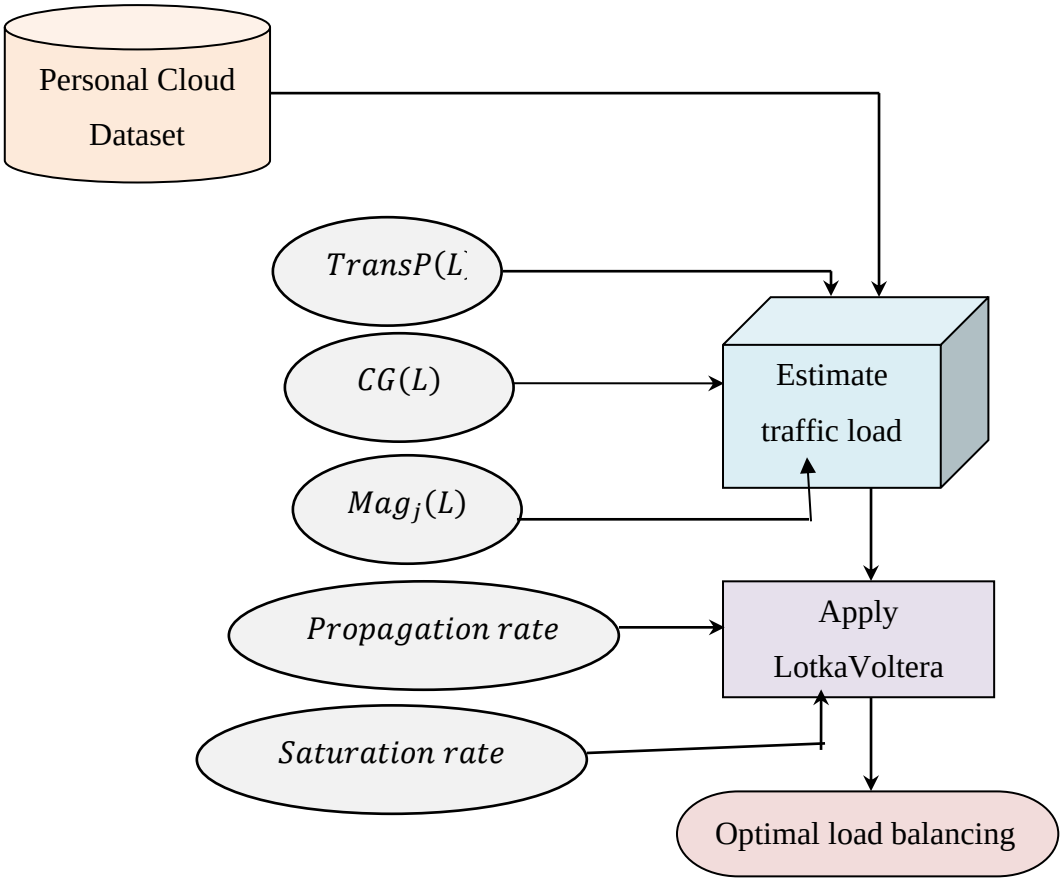

Figure. 2 LotkaVoltera traffic load balancer model

hebbian-context unit of hidden layer matrix. 'CU_{HL}', previously hidden layer. 'HL_{t-1}' and the bias of the hidden layer. ' B_{HL} ' respectively. For the first iteration, the weight is initialized as '1', and at each time instance,the hebbian learning rule is applied to update the importance as given below.

$$\Delta W_i = \eta F_i y \tag{9}$$

$$WM = W = \begin{bmatrix} W_1 \\ W_2 \\ W_3 \\ \dots \\ W_m \end{bmatrix} = \begin{bmatrix} W_{11} & W_{12} & \dots & W_{1n} \\ W_{21} & W_{22} & \dots & W_{2n} \\ W_{31} & W_{32} & \dots & W_{3n} \\ \dots & \dots & \dots & \dots \\ W_{m1} & W_{m2} & \dots & W_{mn} \end{bmatrix} \tag{10}$$

From the above Eqs. (9) and (10), the weights and input features are utilized to generate an output (i.e., resource allocation). The importancegiven above is represented in a matrix called connection matrix. Moreover, in the context unit, cache resource allocation is performed via base stations 'BS'. Here, the total data or packets acquired by all the users at any time are given below.

$$DP_{Tot} = \sum_{i=1}^{n} DP_i \tag{11}$$

For each cache level resource allocation, to increase the total receivable data packets of all users is formulated as given below.

$$DP_i = MAX\left[\sum_{i=1}^{m}\sum_{j=1}^{n} \varphi_{ij} F_j\right], such\ that\ \frac{\varphi_{ij} F_j}{\vartheta} \leq LC_{ij}^{CI} \tag{12}$$

From the above Eq. (12), 'φ_{ij}' represent the optimal cache resource allocation factor concerning the given input features. 'F_j'. This total receivable data packets of all users 'DP_i' is formulated in such a manner that the transmission delay 'ϑ' represents the aggregated sum of data packet processing and data packet exchange time between user-requested tasks 'T' and the base stations 'BS'. Finally, the result is obtained in the output layer, as given below.

$$OL_t = \sigma_{OL}(W_{OL} HL_t + B_{OL}) \tag{13}$$

From the above Eq. (13), the output layer. 'OL_t' is formulated utilizing the output layer activation

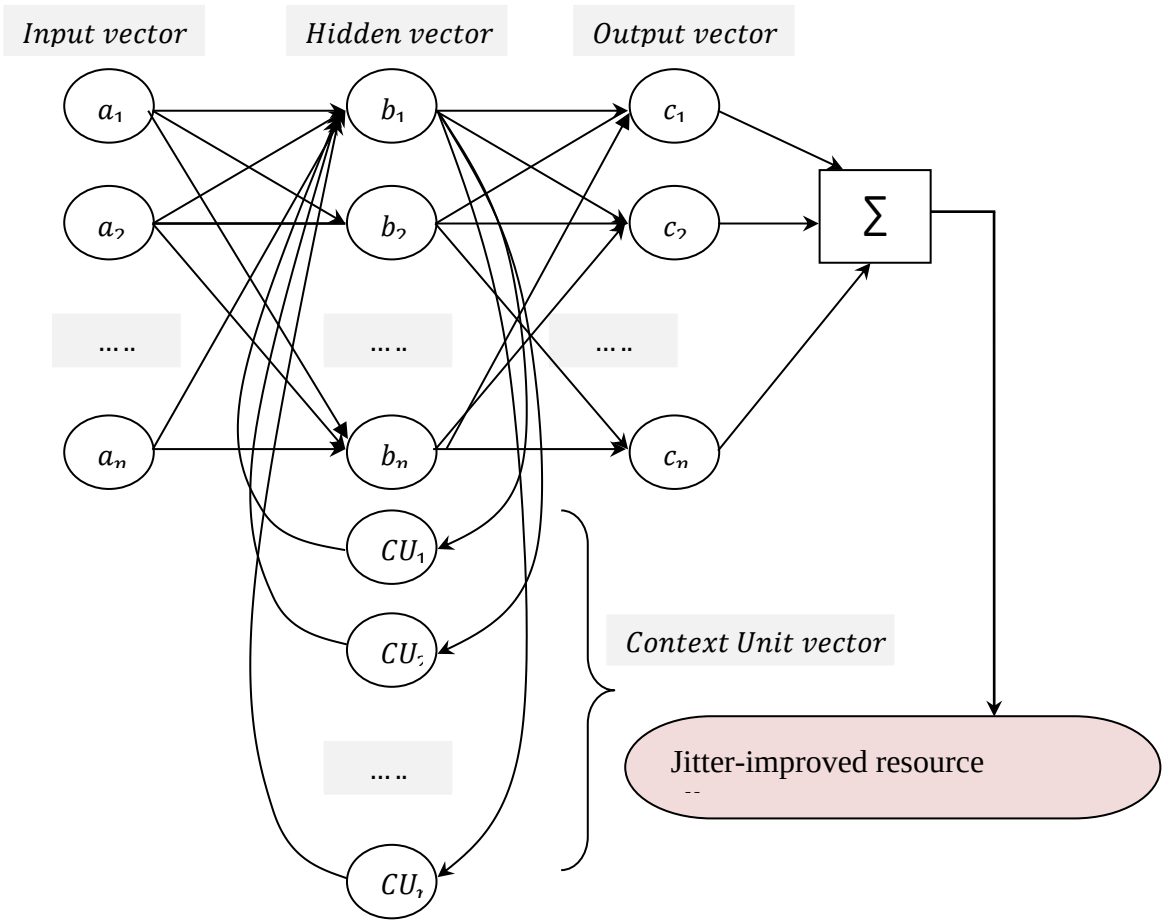

Figure. 3 Structure of the input layer and hidden layer, and an output layer

function. 'σ_{OL}',the weight of the output layer matrix 'W_{OL}' hidden layer 'HL_t' and the bias of the output layer. ' B_{OL} ' respectively. The pseudo-code representati on of elman hebbian-recurrent neural network-based cache resource allocation is given below.

In the above elman hebbian-recurrent neural network-based cache resource allocation algorithm, the input layer receives feature values and load-balanced virtual machines for each user request. Then, the hidden layer formulates the resource allocation problem—hebbian based context unit weight update. Also, user-requested tasks and base stations exchange cache-level data packets. Finally, the output layer shows whether resources were allocated. Because the cache is handled instead of the cloud data centre, the delay between received data packets is minimized. This reduces jitter and increases the makespan.

Table 1. Storage layer description

S. No	Feature	Description
1	Volume	Considered a directory with three types of volumes: i) root/predefined, ii) UDF (user-defined folder), and iii) share (sub-volume of another user to which the current user has access).
2	Node	A node is a file or a directory in the system.
3	Session	The session is used to identify requests of a single user during a session lifetime that does not expire automatically. The client may disconnect, or the server may go down, therefore resulting at the end of the session
4	Request types	There are different request types. They are storage, session and RPC.

4. Experimental settings

In this section, iFogSim simulates the proposed LV-EHRCC and existing methods, namely EPRAM[1] and load balancing for FoT-Gateways [2] that is investigated over different performance metrics using personal cloud dataset. (obtained from http://cloudspaces.eu/results/datasets). Ifogsim simulator measure load balancing and resource management across for and cloud resource.

Table 2. Column fields description

S. No	Feature	Description
1	Row_id	Database row identifier
2	Account_id	Personal cloud account
3	File_size	Size of the uploaded file
4	Operation_time_start	Starting time of the API call
5	Operation_time_end	Ending time of the API call
6	Time_zone	Time zone of a node
7	Operation_ID	The ID of the API call
8	Operation_Type	Type of API call
9	Bandwidth_Trace	Time series trace
10	Node_ip	The IP address of the node
11	Node_name	Name of node
12	Quota_start	Amount of data at the start of API call
13	Quota_end	Amount of data at the end of API call
14	Quota_total	The total amount of data
15	Capped	Capped or not
16	Failed	Indicates if API has failed
17	Failure	Available failure information

Table 3. Comparison results of load balancing efficiency using LV-EHRCC, EPRAM [1] and load balancing for FoT [2]

Number of tasks	Load balancing efficiency (%)		
	LV-EHRCC	EPRAM	Load balancing for FoT
5000	98.7	97.7	96.3
10000	97.15	94.35	92.15
15000	97	94.15	90
20000	96.85	93.56	88.35
25000	96.35	92.15	86.15
30000	96	91.15	83.25
35000	95.15	90.35	81
40000	95.05	87.35	80.15
45000	94.35	85.25	79.35
50000	93.25	83	78

4.1 Dataset details

The NEC personal cloud dataset integrates two sources of information, i.e., from the storage layer and sharing interactions. Table 1 below provides the storage layer description, whereas Table 2 below provides the column field description.

5. Discussion

This section compares the proposed (LV-EHRCC) in an FC environment with existing state-of-the-art methods, EPRAM [1] and load balancing for FoT-Gateways [2]. In addition, personal cloud datasets are utilized for performance analysis using metrics, Bandwidth, load balancing efficiency, jitter, and makespan for 50000 specific user-requested jobs in the iFogSim simulator using graphical user interfaces.

5.1 Performance analysis of load balancing

Load balancing distributes network traffic between servers to improve potential and dependability. It also refers to user-requested task dispersal. This efficient workload distribution manages the workload better by allocating resources between numerous servers quickly, resulting in higher performance. The goal is to distribute load among multiple servers effectively. When one server is busy with user requests and others are idle, some burdens are moved to a nearby server with fewer requests. By distributing workload, Bandwidth and jitter are effectively used. This is shown below.

$$Eff_{LB} = \left[\frac{T_{AE}}{n}\right] X100 \tag{14}$$

From the above Eq. (14), load balancing efficiency. 'Eff_{LB}' is measured based on the user request tasks allocated efficiently. 'T_{AE}' to the total number of user-requested tasks 'n' in the queue. The load balancing efficiency is measured in terms of percentage (%). Table 3 below lists the load balancing efficiency results obtained using the equation from (14).

Load balancing efficiency measures how well user-requested activities are assigned in fog computing. Fig. 4 shows load balancing efficiency for LV-EHRCC, EPRAM [1], and FoT [2] at different service sizes (i.e., concerning other numbers of tasks in the range of 5000 to 50000). According to the analysis of the results, the difference between the three approaches is minimal but grows when the number of user-requested actions increases. Our load-balancing approach performs better for all task sizes. The simulation with 5000 tasks showed that the three techniques were 98.7%, 97.7%, and 96.3 & efficient at load balancing. The suggested LV-EHRCC approach balances load more efficiently

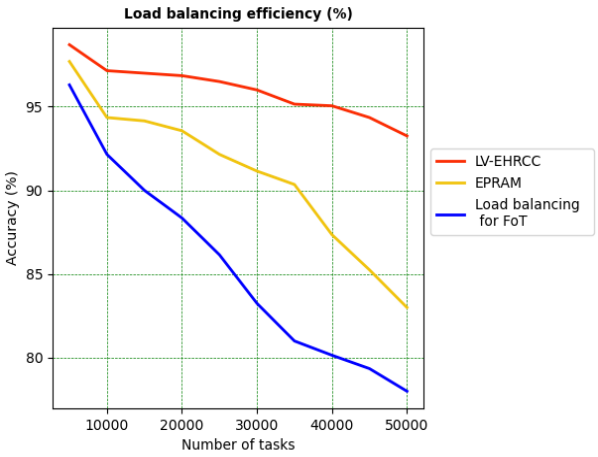

Figure. 4 Graphical representation of load balancing efficiency

Table 4. Comparison results of Bandwidth using LV-EHRCC, EPRAM [1] and load balancing for FoT [2]

Number of tasks	Bandwidth (Mbps)		
	LV-EHRCC	EPRAM	Load balancing for FoT
5000	20	21	18
10000	23	22	19
15000	25	24	20
20000	28	27	22
25000	31	28	23
30000	35	31	25
35000	38	33	27
40000	40	35	29
45000	42	38	32
50000	45	40	35

than [1] and [2]. LotkaVoltera's optimization function boosted performance. This function examined propagation speed, task saturation rate, and queued tasks. Thus, VMs were balanced and allocated to user-requested tasks. The LSR LV-EHRCC approach improves load balancing efficiency by 6% compared to [1] and 13% compared to [2].

5.2 Performance analysis of Bandwidth

Bandwidth is the amount of data sent at one time and the most significant quantity of data or packets transmitted over the internet in a given time. It relates to network speed, not data packet movement speed. More data packets are delivered with higher Bandwidth. Bandwidth is the quantity of data that can be transferred in a given period within a network. Table 4 includes bandwidth measurements utilizing LV-EHRCC, EPRAM [1], and Load balancing for FoT [2].

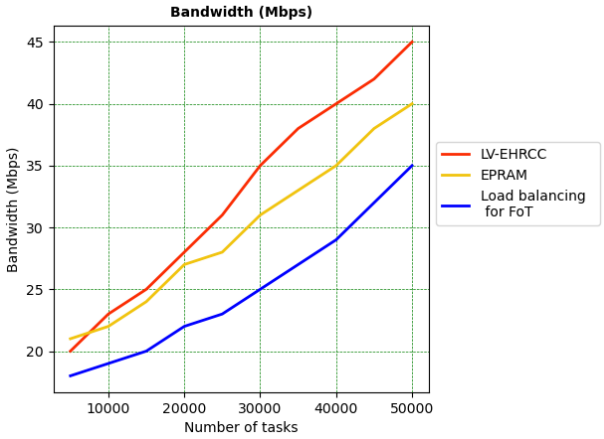

Figure. 5 Graphical representation of Bandwidth

Table 5. Comparison results of load balancing efficiency using LV-EHRCC, EPRAM [1] and load balancing for FoT [2]

Number of tasks	Makespan (ms)		
	LV-EHRCC	EPRAM	Load balancing for FoT
5000	7.5	12.5	16.5
10000	9.35	12.85	17.25
15000	10.15	13.35	17.85
20000	10.85	14	18.35
25000	11.25	15.15	18.85
30000	11.85	17.85	21.35
35000	12.35	19.35	25.25
40000	14.55	20.15	27.15
45000	16	23.45	29
50000	17.85	25	29.85

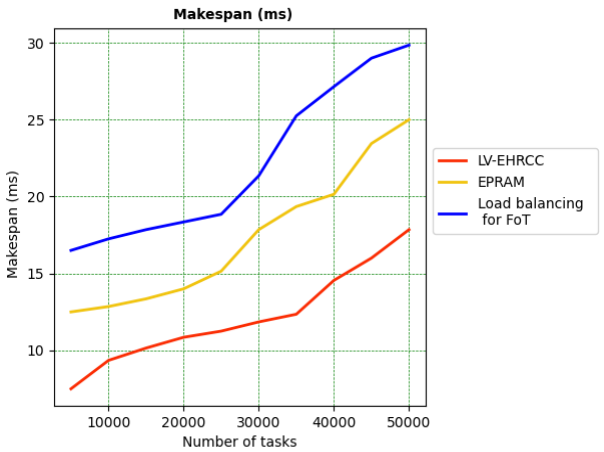

Figure. 6 Graphical representation of makespan

5.3 Performance analysis of makespan

Makespan refers to the time difference between

the beginning and finishing of a sequence of user-requested tasks. This is obtained as given below.

$$Ms=n^* t\ [SOT] \quad (15)$$

From the above Eq. (15), makespan 'Ms' denotes the number of user-requested tasks waiting in the queue 'n' and the time involved in scheduling a single user-requested task 't [SOT]'. It is measured in terms of milliseconds (ms).

Table 5 below lists the makespan efficiency results obtained using the equation from (15).

Fig. 6, given above, shows the impact of makespan for 50000 different user-requested tasks involved in optimal resource allocation in a fog computing environment. For other numbers of tasks, the makespan also varies, with no uniformity observed between the three methods. However, with simulations involving 5000 tasks, the time involved in scheduling a single user-requested study using the three methods was found to be 0.0015ms, 0.0025ms and 0.0033ms, respectively. With this, the overall makespan for 5000 tasks was observed to be 7.5ms, 12.5ms and 16.5ms using LV-EHRCC, EPRAM [1] and load balancing for FoT [2], respectively, showing an improvement using the proposed LV-EHRCC method. The progress was the application of the Elman Recurrent Neural Network. By applying this, cache resource allocation was performed in the context units via Base Stations 'BS'. Due to this optimal resource allocations were ensured by obtaining total receivable data packets of all users in such a way by reducing the transmission delay and data packet exchange time between user-requested tasks and the Base Stations 'BS'. With this the makespan using the proposed LV-EHRCC method was found to be 30% improved compared to [1] and 45% improved compared to [2].

5.4 Performance analysis of jitter

Finally, the jitter is the delay between received data packets. Jitter is considered the dissimilarity in the data packet flow between two systems that might occur owing to network congestion. With the application of the FC environment, the jitter can be reduced considerably. The jitter is measured by identifying the average time difference between each packet sequence.

$$J = (DPSeq_{TD})/n \quad (16)$$

Table 6. Comparison results of load balancing efficiency using LV-EHRCC, EPRAM [1] and load balancing for FoT [2]

Number of tasks	**Jitter (ms)**		
	LV-EHRCC	EPRAM	Load balancing for FoT
5000	7.8	10.2	12.6
10000	9.5	11.35	16.15
15000	12.35	15.15	18
20000	14.15	18.25	22.55
25000	15.85	21.35	27.35
30000	17	22.15	29.25
35000	18.35	24.35	31.35
40000	21.45	28.15	33
45000	28.35	31.35	35.15
50000	30	34	37

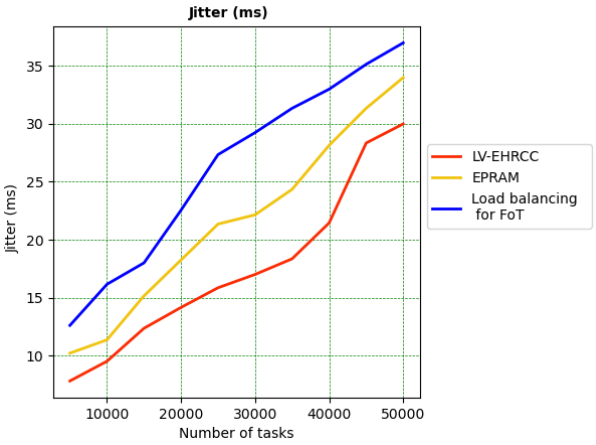

Figure. 7 Graphical representation of jitter

Table 6 below lists the jitter measure results obtained using the equation from (16).

Fig. 7 displays the ideal FC resource allocation jitter rate. The figure shows that all three approaches increased jitter for 25000 tasks. For 5000 user-requested activities, 5 data packet sequence flows, the time difference using the suggested LV-EHRCC technique was 12ms, 11ms, 8ms, 3ms, 5ms, 15ms, 13ms, 11ms, 5ms, 7ms, and 18ms, 14ms, 13ms, 8ms, 10ms. The three approaches' jitter rates were 7.8ms, 10.2ms, and 12.6ms. LV-EHRCC had a lower jitter rate than [1, 2]. Elman hebbian-recurrent neural network-based cache resource allocation algorithm minimized jitter rate. Our technique used a hebbian learning-based context unit weight update in the hidden layer to pose the resource allocation problem. In the case of resource requirements, the cache was processed instead of directly retrieving it from the cloud data centre. This reduces the LV-EHRCC jitter rate by 20% vs [1] and 35% vs [2].

6. Conclusion

This study discusses fog computing resource allocation. Lotka Voltera load balancer and elman hebbian-recurrent neural network cache resource allocation (LV-EHRCC) are presented for fog computing. LV-EHRCC approach balances the load and allocates appropriate resources in fog computing. The first bandwidth-efficient Lotka Voltera traffic load balancer model balances incoming loads among virtual machines. Elman hebbian-recurrent neural network-based cache resource allocation approach improves jitter-optimal resource allocation. Simulations indicated that LV-EHRCC enhances Bandwidth, jitter rate, and load balancing efficiency over state-of-the-art techniques.

Conflicts of interest

The authors declare no conflict of interest. Authors declare that any personal circumstances or interest may be perceived as inappropriately influencing the representation or interpretation of reported research results.

Author contributions

The contributions of authors are as follows:

S.V. Nethaji: Conceptualization, Methodology, software, validation, formal analysis, investigation, data curation and writing-original paper draft. Dr. M. Chidambaram: Validation, supervision.

References

[1] F. M. Talaat, "Effective prediction and resource allocation method (EPRAM) in fog computing environment for smart healthcare system", *Multimedia Tools and Applications*, Vol. 81, No. 6, pp. 8235-8258, 2022.

[2] E. Batista, G. Figueiredo, and C. Prazeres, "Load balancing between fog and cloud in fog of things based platforms through software-defined networking", *Journal of King Saud University-Computer and Information Sciences*, Vol. 34, No. 9, pp. 7111-7125, 2022.

[3] N. Mostafa, "Resource Selection Service Based on Neural Network in Fog Environment", *Technology and Engineering Systems Journal*, Vol. 5, pp. 408-417, 2020.

[4] H. T. Dang, S. Bhardwaj, T. Rahim, A. Musaddiq, and D. S. Kim, "Reinforcement learning based resource management for fog computing environment: Literature review, challenges, and open issues", *Journal of Communications and Networks*, Vol. 24, No. 1, pp. 83-98, 2022.

[5] M. S. Sofla, M. H. Kashani, E. Mahdipour, and R. F. Mirzaee, "Towards effective offloading mechanisms in fog computing", *Multimedia Tools and Applications*, Vol. 81, pp. 1997-2042, 2022.

[6] J. Baek and G. Kaddoum, "Heterogeneous task offloading and resource allocations via deep recurrent reinforcement learning in partial observable multifog networks", *IEEE Internet of Things Journal*, Vol. 8, No. 2, pp. 1041-1056, 2020.

[7] M. Junaid, A. Sohail, R. N. B. Rais, A. Ahmed, O. Khalid, I. A. Khan, and N. Ejaz, "Modeling an optimized approach for load balancing in cloud", *IEEE Access*, Vol. 8, pp. 173208-173226, 2020.

[8] S. F. Abedin, A. K . Bairagi, M. S. Munir, N. H. Tran, and C. S. Hong, "Fog load balancing for massive machine type communications: A game and transport theoretic approach", *IEEE Access*, Vol. 7, pp. 4204-4218, 2018.

[9] C. S. M. Babou, D. Fall, S. Kashihara, Y. Taenaka, M. H. Bhuyan, I. Niang, and Y. Kadobayashi, "Hierarchical load balancing and clustering technique for home edge computing", *IEEE Access*,Vol. 8, pp. 127593-127607, 2020.

[10] M. A. Basset, N. Moustafa, R. Mohamed, O. M. Elkomy, and M. Abouhawwash, "Multiobjective task scheduling approach for fog computing", *IEEE Access*, Vol. 9, pp. 126988-127009, 2021.

[11] S. Smolka and Z. Á. Mann, "Evaluation of fog application placement algorithms: A survey", *Computing*, Vol. 104, pp. 1397–1423, 2022.

[12] B. Jamil, H. Ijaz, M. Shojafar, K. Munir, and R. Buyya, "Resource Allocation and Task Scheduling in Fog Computing and Internet of Everything Environments: A Taxonomy, Review, and Future Directions", *ACM Computing Surveys (CSUR)*, Vol. 54, pp. 1-38, 2022.

[13] J. Shan, "Computing resource allocation strategy considering privacy protection mechanism in edge computing environment", *The Journal of Engineering*, Vol. 2022, pp. 401-410, 2022.

[14] V. Divya and L. R. Sri, "Fault tolerant resource allocation in fog environment using game theory-based reinforcement learning", *Concurrency and Computation Practice and Experience*, Vol. 33, No. 16, pp. 25-33, 2021.

[15] J. Polcari, "An informative interpretation of decision theory: The information theoretic basis for signal-to-noise ratio and log likelihood ratio", *IEEE Access*, Vol. 1, pp. 509-522, 2013.

[16] F. Shi and J. Lin, "Virtual Machine Resource Allocation Optimization in Cloud Computing Based on Multiobjective Genetic Algorithm", *Computational Intelligence and Neuroscience*, Vol. 2022, Article ID 7873131, pp.1-10, 2022.

[17] T. Moulahi, S. E. Khediri, U. Khan, and S. R. Zidi, "A fog computing data reduce level to enhance the cloud of things performance", *International Journal of Communication Systems*, Vol. 34, No. 9, 2021.

[18] Z. Qin, X. Qiu, J Ye, and L. Wang, "User-edge collaborative resource allocation and offloading strategy in edge computing", *Wireless Communications and Mobile Computing*, Vol. 2020, Article ID 8867157, pp. 1-12, 2020.

[19] T. Dlamini and S. Vilakati, "LSTM-based traffic load balancing and resource allocation for an edge system", *Wireless Communications and Mobile Computing*, Vol. 2020, Article ID 8825396, pp. 1-15, 2020.

[20] A. Kaur and B. Kaur, "Load balancing optimization based on hybrid Heuristic-Metaheuristic techniques in cloud environment", *Journal of King Saud University-Computer and Information Sciences*, Vol. 34, No. 3, pp. 813-824, 2019.

[21] S. V. Nethaji and M. Chidambaram, "Resource Optimization in Fog Computing with Shift-Invariant Deep Convolutive Load Balancing", *Webology*, Vol. 18, No. 6, pp. 3845-3861, 2021.

[22] S. V. Nethaji and M. Chidambaram, "Differential Grey Wolf Load-Balanced Stochastic Bellman Deep Reinforced Resource Allocation in Fog Environment", *Applied Computational Intelligence and Soft Computing*, Vol. 2022, Article ID 3183701, pp. 1-13, 2022.