DOI 10.15622/ia.23.6.9

M. ELLAKKIYA, T. RAVI, S. PANNEER AROKIARAJ

RUZICKA INDEXIVE THROTTLED DEEP NEURAL LEARNING FOR RESOURCE-EFFICIENT LOAD BALANCING IN A CLOUD ENVIRONMENT

Ellakkiya M., Ravi T., Panneer Arokiaraj S. **Ruzicka Indexive Throttled Deep Neural Learning for Resource-Efficient Load Balancing in a Cloud Environment.**

Abstract. Cloud Computing (CC) is a prominent technology that permits users as well as organizations to access services based on their requirements. This computing method presents storage, deployment platforms, as well as suitable access to web services over the internet. Load balancing is a crucial factor for optimizing computing and storage. It aims to dispense workload across every virtual machine in a reasonable manner. Several load balancing techniques have been conventionally developed and are available in the literature. However, achieving efficient load balancing with minimal makespan and improved throughput remains a challenging issue. To enhance load balancing efficiency, a novel technique called Ruzicka Indexive Throttle Load Balanced Deep Neural Learning (RITLBDNL) is designed. The primary objective of RITLBDNL is to enhance throughput and minimize the makespan in the cloud. In the RITLBDNL technique, a deep neural learning model contains one input layer, two hidden layers, as well as one output layer to enhance load balancing performance. In the input layer, the number of cloud user tasks is collected and sent to hidden layer 1. In that layer, the load balancer in the cloud server analyzes the virtual machine resource status depending on energy, bandwidth, memory, and CPU using the Ruzicka Similarity Index. Then, it is classified VMs as overloaded, less loaded, or balanced. The analysis results are then transmitted to hidden layer 2, where Throttled Load Balancing is performed to dispense the workload of weighty loaded virtual machines to minimum loaded ones. The cloud server efficiently balances the workload between the virtual machines in higher throughput and lower response time and makespan for handling a huge number of incoming tasks. To evaluate experiments, the proposed technique is compared with other existing load balancing methods. The result shows that the proposed RITLBDNL provides better performance of higher load balancing efficiency of 7%, throughput of 46% lesser makespan of 41%, and response time of 28% than compared to conventional methods.

Keywords: cloud computing, load balancing, deep learning, Ruzicka similarity index, throttled load balancing.

1. Introduction. CC is a paradigm that includes distributing services and resources more than the internet. Load balancing (LB) in CC is a significant aspect that is a pivotal task in optimizing resource utilization, enhancing performance, and ensuring high availability of applications and services. As cloud environments consist of multiple servers and resources, distributing incoming user requests efficiently among virtual machines becomes essential to prevent the overloading of any single machine in the cloud server. This distribution of workload helps in achieving optimal resource consumption, enhancing the efficiency of applications.

Task Scheduling- DT (TS-DT) method was developed [1] to distribute and execute tasks within an application. The algorithm

successfully enhances load balancing and reduces makespan but it failed in achieving energy-aware load balancing with minimal response time. A P2BED-C was developed in [2] to minimize energy consumption. However, the efficiency of the method was not improved.

A Reinforcement Learning (RL) model was developed in [3] to optimize cloud resource utilization for providing the best Quality of Service (QoS). However, the makespan was not efficiently reduced. A Dynamic and Resource-Aware Load Balancing technique was introduced [4] to enhance throughput and reduce makespan. However, a resource-aware scheduling approach was not employed for the distribution of tasks on virtual machines (VMs).

The Predictive Priority-based Modified Heterogeneous algorithm was designed in [5] to achieve efficient and dynamic resource provisioning for end user's requirements. However, it did not implement a more effective resource provisioning scheme for end-users. The Bio-Inspired Improved Lion Optimization method was designed in [6], to address load balancing issues through enhancing throughput as well as reducing migration time. However, the performance of efficiency remained unaddressed.

A content-aware machine learning technique was introduced in [7] for enhancing load balancing, leading to improved throughput and minimized response time. However, failed to reduce migration time. Dynamic load balancing method was developed in [8] by Q-learning for resource allocation, resource accessibility, and consideration of user preferences with the aim of minimizing response time and resource consumption. However, it did not achieve higher efficiency in a multitasking environment.

In [9], a multi-objective task scheduling technique was designed with the aim of optimizing scheduling, increasing throughput, as well as reducing both makespan and resource utilization. However, it did not address the minimization of response time. A dynamic virtual machine consolidation method was introduced in [10] for LB to mitigate tradeoffs among energy utilization as well as time complexity.

1.1. Contributions in this article are as follows. The main contributions of the paper as given below.

To enhance load balancing efficiency, the RITLBDNL technique has been developed by Deep Neural Learning and Throttled Load Balancing.

The RITLBDNL technique utilizes the Ruzicka Similarity Index to analyze incoming user tasks and determine the resource status of VMs.

The Throttled Load Balancing process is applied to deep neural learning for task migration from heavily loaded virtual machines to less loaded virtual machines with higher efficiency.

Finally, comprehensive and comparative experiments have been conducted to perform quantitative analysis using various performance metrics.

1.2. Paper organization. The remainder of this article is organized into dissimilar sections: Section 2 explains the literature survey. Section 3 presents the RITLBDNL method. Section 4 details the experimental analysis and describes the dataset. Section 5 gives a performance assessment of the proposed algorithm in comparison to conventional techniques. At last, section 6 gives conclusions of the paper.

2. Literature survey. Load Balancing Protocol was developed in [11] for CC with the aim of minimizing Makespan as well as throughput of VM utilization. Long Short-Term Memory Networks (LSTM) Machine Learning (ML) algorithm was designed in [12] for enhancing load balancing through optimized resource allocation. However, it did not succeed in enhancing the system performance of LB. An integrated optimization algorithm was developed in [13] to make an effective load balancing system that guarantees resource utilization and minimizes task response time.

Component-based throttled load balancing method was introduced in [14], but it failed to consider additional parameters for ensuring the optimal performance of load balancing algorithms. The Receiver-Initiated Deadline-Aware LB approach was developed in [15], and aimed to facilitate migration of incoming tasks to suitable virtual machines. However, this approach was not employed for scientific workflow applications for diverse QoS parameters.

An Action-Based Load Balancing scheme was designed [16] with the aim of reducing makespan and optimizing resource utilization. However, it failed to address resource allocation and management concepts within a cloud data center. A new resource optimization framework was introduced in [17] specifically designed for achieving load balancing with minimal resource utilization. An optimal load balancing method was developed [18], which effectively balances the load on cloud servers.

A re-modified throttled algorithm was developed in [19] to minimize the risk of load imbalance by considering the availability of VMs. However, it failed to address the issues related to increasing the efficiency of the algorithm. A load balancing approach based on renewable energy was developed in [20] to optimize interactive task allocation, aiming to reduce energy costs.

Modified honeybee behavior load balancing (HBB-LB) was introduced in [21] to secure the cloud system. However, the system performance was not enhanced. The Sine Cosine-based Elephant Herding

Optimization (SCEHO) algorithm was combined in [22] by Improved Particle Swarm Optimization (IPSO). Task scheduling behavior was improved but, throughput was not increased.

The two-stage genetic mechanism was utilized in [23] to monitor and manage VMH. But, it failed to minimize the time. A deep load balancer was introduced in [24] to allocate resources with less delay. Nevertheless, it failed to enhance throughput. Improved Lion Optimization (ILO) with Min-Max Algorithm was developed in [25] to identify the optimum solution. However, the load balancing efficiency was not sufficient.

3. Proposal methodology. In cloud computing, dynamically provisioning the resources for applications is a key and challenging task. However, cloud providers face resource management concerns due to inconsistent workloads in heterogeneous environments. The cloud service provider focuses on resource consumption, while the cloud user aims to achieve a shorter makespan time. Therefore, achieving load balancing is a significant parameter for effective task execution to obtain optimal consumption of cloud resources. A new RITLBDNL method is developed for efficient load balancing in a cloud computing environment. Figure 1 depicts a diagram of the RITLBDNL method for efficient LB in the cloud.

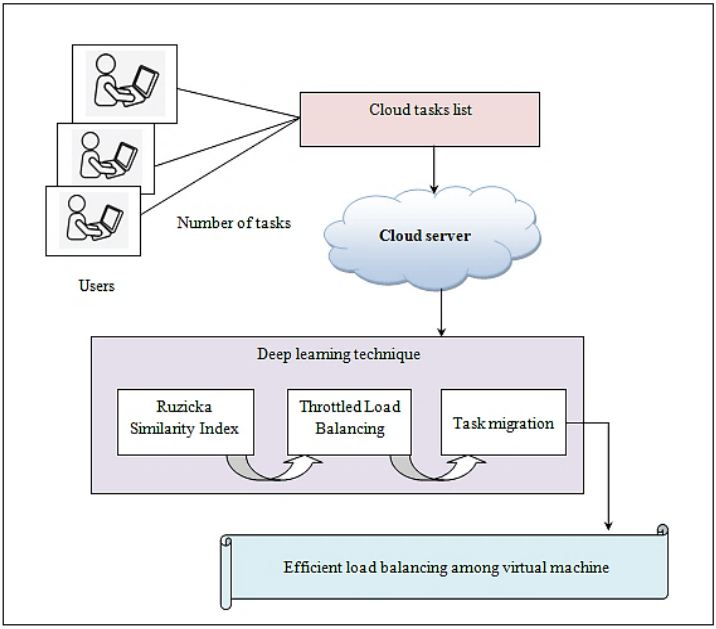

Fig. 1. Architecture diagram of the proposed RITLBDNL technique

Figure 1 demonstrates the RITLBDNL technique uses the deep learning concept for efficient load balancing in the cloud. The four components as cloud user (U), cloud server (CS), load balancer (LB), and virtual machines (V_m) included in the above figure. The working mechanism of the RITLBDNL model uses deep neural learning with several layers. The technique collects the number of cloud user requests or tasks. Ruzicka Similarity Index is utilized in hidden layer 1 to examine the virtual machine resource status. In hidden layer 2, the workload from heavily loaded virtual machines to less loaded ones is distributed to perform task migration by Throttled Load Balancing. In this way, throughput is improved and response time and makespan are minimized.

3.1. System model. It involves four key entities namely cloud user (U), cloud server (CS), load balancer (LB), and virtual machines (V_m). Initially, the cloud user 'U' submits numerous tasks, denoted as $T = \{T_1, T_2, \dots, T_n\}$, to the cloud server (CS). CS receives these tasks as of U. Subsequently, the load balancer within the cloud server analyzes and determines the status of virtual machines, categorizing them as minimum loaded, overloaded, as well as balanced load capacity. Once VM statuses are identified, the load balancer executes task migration using throttled load balancing with higher efficiency.

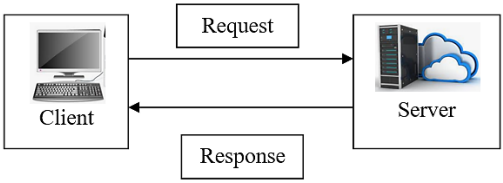

Fig. 2. Overview model of the client-server system

In above Figure 2, the client-server model includes the Server or Client. The client-server model explains the communication among two computing entities over a network. A client is a program that creates requests to a server. A server is a program that responds to those requests.

3.2. Ruzicka Indexive Throttle Load Balanced Deep Neural Learning. Deep Learning (DL) is a type of ML, which focuses on the development, and training of Artificial Neural Network (ANN) to perform some process. The term "deep" refers to the use of Deep Neural Networks (DNNs), which contain numerous hidden layers among input as well as output layers. These networks are referred to as DNNs or DL models.

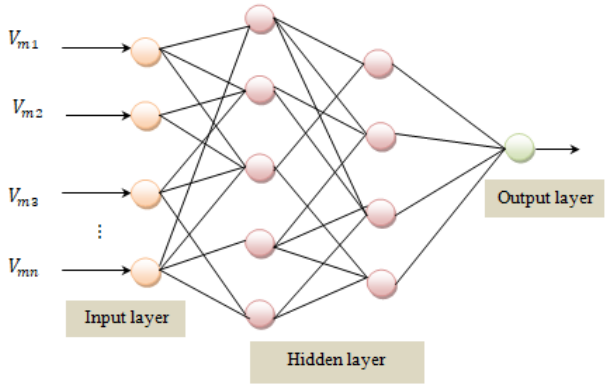

Fig. 3. Structures of the deep neural networks

Figure 3 depicts the structure of DNNs. The DNN is a fully connected feed-forward artificial neural network and it generates a set of outputs from a set of inputs. A DNN is constructed with three main layers such as input, hidden (i.e. middle), and output layers. The input and output layers are always single layers, whereas the middle layer includes two sublayers for analyzing the given input. Each layer is typically composed of small individual units called artificial neurons or nodes. The artificial neuron has the ability to process the given weighted inputs and applies an activation function and forward output to other nodes in the network. An input to an artificial neuron is a number of virtual machines (Vm_i). The neuron in one layer is fully connected to the neuron in another layer.

Each connection between neurons has an associated weight, which determines the strength of the connection. It also has an associated bias. The equation for a single neuron is expressed mathematically as follows:

$$Y = \mathrm{F}[\mathrm{S}], \tag{1}$$

$$\mathrm{S}=\sum_{i=1}^{n} (Vm_i * w_{ij}) + Q, \tag{2}$$

where Y indicates an output of the neuron, w_{ij} denotes the weight of the connection between the i^{th} neuron in the previous layer and the j^{th} neuron in the current layer, $Vm_i * w_{ij}$ denotes a product of the weight (w_{ij}) associated with the connection between the i^{th} neuron in the previous layer and j^{th} neuron in the current layer, and the input (Vm_i i.e $virtual\ machine$) from the i^{th} neuron in the previous layer. From

(2), 'Q' indicates a bias term that stores the numeric value as one, F denotes a sigmoid activation function used to determine whether a neuron is activated or not, it suggests that the neuron output is binary, typically representing a binary classification decision (activated or not activated).

$$\mathrm{F}\,[\mathrm{S}] = \frac{1}{1+\exp(-\mathrm{S})}, \tag{3}$$

where F $[S]$neuron's output with sigmoid activation is passed to the next layer of neurons

The input is transferred to a hidden layer where the resource availability of a virtual machine is determined to facilitate efficient load balancing.

$$RA\,(Vm_i) = \{Mem_c, BaW_C, E_C, CPU_{Ut}\}, \tag{4}$$

where $RA\,(Vm_i)$ denotes a resource's availability of the virtual machine that includes a memory capacity 'Mem_c', bandwidth capacity 'BaW_C', energy capacity 'E_C' and CPU utilization 'CPU_{Ut}'.

Initially, the memory capacity is determined by calculating the variance among total available memory as well as utilized memory.

$$Mem_c = To_{Memc} - Con_{Memc}, \tag{5}$$

where Mem_c indicates the memory capacity of the VM and To_{Memc} indicates the total memory capacity, Con_{Memc} represents the utilized memory capacity. Variation between the total memory capacity and the utilized memory capacity measurement is employed to assess the present memory status of the VM.

The bandwidth of a virtual machine denotes its capability to handle the maximum amount of data, typically measured in Mbps (megabits per second). The current status of bandwidth is determined through mathematical calculations.

$$BaW_C = Ba_{VM\,(total)} - Ba_{VM\,(con)}, \tag{6}$$

where BaW_C denotes the bandwidth capacity, $Ba_{VM\,(total)}$ indicates the total bandwidth, and $Ba_{VM\,(con)}$ represents the utilized bandwidth. Depending on the above-said metrics, the current status of the bandwidth capacity is determined.

The total energy consumption is calculated depending on the energy usage of the VM. Energy utilization is measured in kWh. Thus, the energy capacity of the virtual machine is determined as follows:

$$E_C = [Tot_E] - [Con_E], \tag{7}$$

where E_C represents the energy capacity, Tot_E indicates the total energy, Con_E denotes the consumed energy.

The CPU utilization time of the VM is computed mathematically by calculating the variance between the total time and the time spent processing specific tasks. This calculation helps to assess the efficiency and resource consumption of the virtual machine during the execution of its assigned workload.

$$CPU_{Ut} = [T_{cpu}] - [C_{cpu}], \tag{8}$$

where CPU_{Ut} denotes the CPU time, T_{cpu} indicates the total time and C_{cpu}symbolizes the consumed time of VM.

The proposed RITLBDNL technique finds the resource availability of a virtual machine based on the energy, bandwidth, memory, and CPU through the similarity measure. Ruzicka Similarity Index is employed for discovering the similarity between two sets. It provides a range from 0 to 1. Ruzicka Similarity Index is used to analyze the VM resource status as well as categorize VM as OL, minimum loaded and BL. The mathematical formula for calculating the similarity between the nodes is shown below

$$\beta = \frac{[RA\ (Vm_i)\ \cap\ T)\ ]}{\sum RA\ (Vm_i)\ + \sum T - [RA\ (Vm_i)\ \cap\ T)\ ]}, \tag{9}$$

where 'β' denotes a Ruzicka similarity coefficient, $RA\ (Vm_i)$ denotes the resource availability of the virtual machine and T indicates the threshold (i.e., 0.5),$RA\ (Vm_i)\ \cap\ T$ denotes a mutual dependence between the resource availability and threshold. The coefficient (β) provides the output ranges between 0 and 1. Likewise, similarities of all the VMs are computed based on the energy, bandwidth, and memory, and CPU using the statistic similarity coefficient

$$\beta = \begin{cases} < 0.5,\ UL \\ = 0.5\ ,\ BL, \\ > 0.5,\ OL \end{cases} \tag{10}$$

where β denotes the output of coefficient. Depend on coefficient outcome, LB determines over loaded (OL) , under loaded (UL) and balanced load (BL).

Throttled Load Balancing refers to a type of load balancing mechanism that includes throttling. Load balancing is the process of dispensing network traffic or calculating workload across numerous resources to guarantee no one, virtual node is overloaded. Throttling, in this context, involves controlling the rate at which certain requests are processed to manage the load on the system.

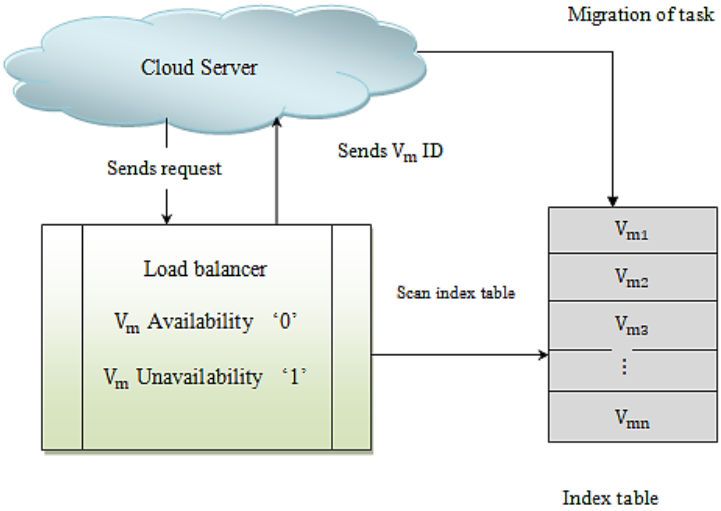

Fig. 4. Flow Process of throttled load balancing

Figure 4 depicts the flow process of throttled load balancing that contains cloud server, LB and several V_m $V_{m1}, V_{m2}, V_{m3}, \ldots V_{mn}$. Initially, a number of tasks are sent to CS. Then the server sends requests to the load balancer to identify the accessibility of the VM.

$$CS \overset{req}{\Longrightarrow} LB, \tag{11}$$

where req denotes the request. After receiving the request, the load balancer maintains a complete list of virtual machines using an index table and responds with the availability status.

$$\text{LB} \rightarrow \begin{cases} 1;\ V_m'sunavailable \\ 0;\ V_m'savailable \end{cases}, \tag{12}$$

where the status of $V_m's$ is identified through '1' and '0'.After that, the LB starts to scan index tables and send the less-loaded and heavily loaded VM IDs' to the cloud server. The server performs tasks migration from a heavily loaded to a less loaded virtual machine. In this way, resource-efficient load balancing is obtained at the output layer. The algorithm of Ruzicka indexive Throttle Load Balanced Deep neural learning is given below.

//Algorithm 1: Ruzicka indexive Throttle Load Balanced Deep neural learning

Input: Number of cloud user requests $T_1, T_2, T_3, \ldots, T_n$; virtual machines $(Vm_1, Vm_2, \ldots, Vm_n)$; cloud server (CS), loads balancer (LB),

Output: Increase the load balancing efficiency

Begin

1. Send number of requests or tasks $T_1, T_2, T_3, \ldots, T_n$ to CS
2. LB find the resoure capcity of virtual machine 'Vm' **—hidden layer 1**
3. **For each** virtual machine 'Vm'
4. Compute the multiple resources '$RA\ (Vm_i)$' using (5) (6) (7) (8)
5. Measure the Ruzicka similarity coefficient 'β'
6. ***if*** $(\beta > 0.5)$ **then**
7. virtual machine is classified as a overloaded
8. ***elseif*** $(\beta = 0.5)$ **then**
9. virtual machine is classified as a balanced loaded
10. ***elseif*** $(\beta < 0.5)$ **then**
11. virtual machine is classified as a less loaded
12. **End if**
13. CS send request to LB **---hidden layer**
14. LB sends virtual machine availability status to server
15. LB scan the index table
16. LB sends the less loaded and overloaded virtual machine ID's to server
17. CS perform task migration from overloaded to less loaded virtual machine
18. Obtain final load balancing **at the output layer**
19. **End for**

End

Algorithm 1 described above outlines the process of load balancing by the Ruzicka Indexive Throttle Load-Balanced Deep Neural Learning approach. For each incoming task from the user, the load balancer in the cloud server estimates the resource availability of the VM by Ruzicka Index function. This function is utilized to calculate the load status of each VM in the first hidden layer, classifying them as less loaded, overloaded, and balanced loaded. Subsequently, LB transmits the IDs of the minimum loaded and overloaded VMs to the cloud server. The server then makes a decision regarding the immigration of tasks from the overloaded VM to the less loaded one, focusing on the second hidden layer of deep learning techniques. As a result, the cloud server efficiently balances the workload between VMs with minimal time. This approach proves beneficial in managing a huge number of incoming tasks, leading to minimization of makespan and an increase in throughput.

4. Experimental setup. Experimental evaluation of RITLBDNL and conventional methods, such as TS-DT [1], P2BED-C [2], and RL Approach [3] are implemented using the Java language. To conduct the experiment, we utilize the Personal Cloud Dataset obtained from http://cloudspaces.eu/results/datasets. Major intend of the dataset is to facilitate load balancing. It contains 17 attributes, and 66,245 instances. 17 attributes are row id, account id, file size (task size), operation_time_start, and so on. Two columns, namely time zone and capped, are excluded from the analysis. The aforementioned columns are selected for the purpose of achieving effective load balancing between numerous VMs by big-data CC

5. Performance Analysis. To estimate the performance of RITLBDNL, a comparative analysis was performed between TS-DT [1], P2BED-C [2], and RL Approach [3] in load balancing efficiency, throughput, makespan, response time and memory consumption in Table 1.

Load balancing efficiency: It refers to the ratio of a number of user requests, which are accurately balanced across all VMs. It is computed as given below:

$$LBE = \left[\frac{correctlybalanceduserrequestes}{n}\right] * 100, \quad (13)$$

where LBE indicates a load balancing efficiency, 'n' denotes the total number of user requests.It is measured in percentage (%).

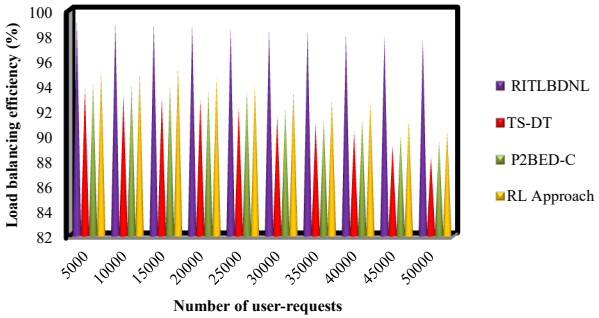

Fig. 5. Analysis of load balancing efficiency

Figure 5 provides a graphical illustration of load balancing efficiency across distinct numbers of user requests ranging from 5,000 to 50,000, taken from the dataset. Figure 4 compares the results of four different algorithms, namely RITLBDNL, TS-DT [1], P2BED-C [2], and RL Approach [3]. It is evident that the RITLBDNL technique yields higher load balancing efficiency. This observation is validated through statistical assessment. In an experiment involving 5000 user requests, the RITLBDNL technique achieved a load balancing efficiency of 99.24%. In contrast, the efficiency of [1], [2], [3] was observed as 93.7%, 94.24%, and 95.2%, respectively. Likewise, different results were attained for every method. Comparing performance outcomes of the proposed method against conventional techniques, overall comparison outcomes show that the RITLBDNL technique increases load balancing efficiency by 8% ,7% and 5% than the [1], [2], [3]. The application of the deep learning technique in RITLBDNL identifies the workload capacity of virtual machines based on resource availability using the Ruzicka Similarity Index function. By utilizing the throttle load-balancing algorithm efficiently, balances workload between VMs, resulting in improved efficiency.

Throughput: it is defined as the ratio of the number of user requests implemented per unit of time in Table 2. It is computed as follows:

$$TP = \left[\frac{Numberofrequestsexecuted}{t\ (seconds)}\right], \tag{14}$$

where 'TP' represents throughput, t indicates time in seconds. It is calculated as requests per second (requests/sec).

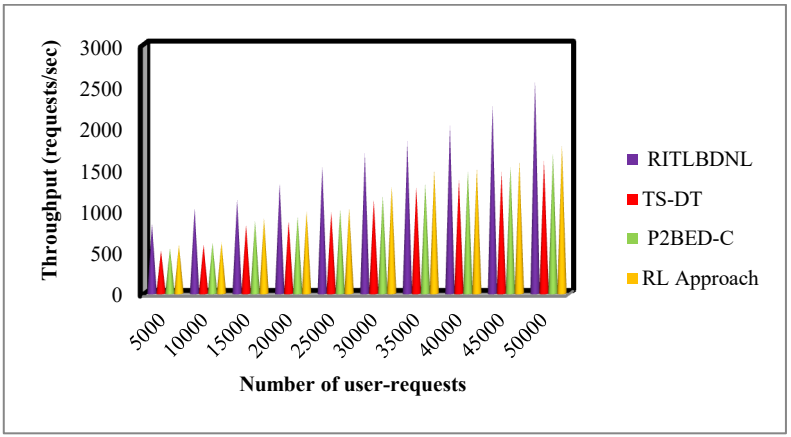

Fig. 6. Analysis of throughput

Figure 6, presented above, illustrates a comparative analysis of throughput. The analysis highlights that the proposed RITLBDNL technique achieved enhanced performance. To ensure the robustness of the RITLBDNL method, ten separate comparisons were conducted for each method. The average of these ten comparisons reveals that the throughput performance using the RITLBDNL technique improved by 54%, 46%, 39% than the [1], [2], [3]. This improvement is achieved through the migration of tasks from the overloaded VMs to the minimum loaded VMs. Consequently, these selected resource-efficient, less loaded virtual machines demonstrate the capability to consistently execute numerous user requests within a specific time.

Makespan: The metric is determined by the duration a virtual machine takes to handle a series of user requests in Table 3. It is calculated as the mathematical dissimilarity among starting as well as completion times of user-requested tasks.

$$M_s = (t_{complete}) - (t_{starting}), \tag{15}$$

where, M_s represents the makespan, $t_{complete}$ indicates request completion time $t_{starting}$de notes a request for finishing time. It is measured in milliseconds (ms).

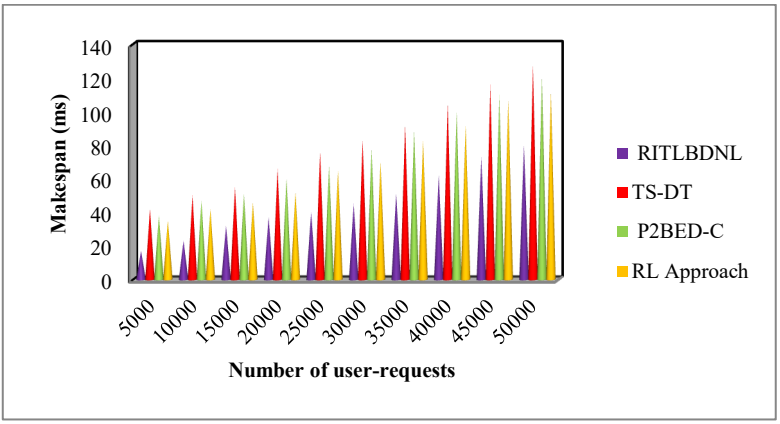

Fig. 7. Analysis of Makespan

Figure 7 depicts a graphical representation of the makespan for load balancing using four methods namely RITLBDNL, TS-DT [1], P2BED-C [2], and RL Approach [3]. The figure illustrates that makespan increases as the number of user requests increases. This occurs as a huge number of user requests during the experiment consumes more time, consequently increasing the makespan. However, in experiments with 5000 user requests, the time taken to complete user requests was only $'17ms'$ using the RITLBDNL technique. The overall makespan was observed to be $42ms$, $38ms$, and $35ms$ for [1], [2], and [3], respectively. Following the experiments, various results were examined for every method. Comprehensive comparison denotes that makespan performance using RITLBDNL is reduced by 45%, 41%, and 36% compared to the existing methods. The RITLBDNL technique employs the Ruzicka similarity index function to analyze the resource status of a VM based on energy, bandwidth, memory, and CPU. Once a minimum loaded VM is identified, LB migrates user requests from an overloaded to a less loaded VM. The less loaded machine requires minimal time to complete the user requests.

Response time: It is defined as the duration it takes to respond user requested tasks in Table 4.

$$RT = n * T\ (\ transmission + waiting + processing), \qquad (16)$$

where RT indicates a response time, n indicates the number of user requests, T represents time for broadcasting, waiting, and processing the user requests. It is measured in milliseconds (ms).

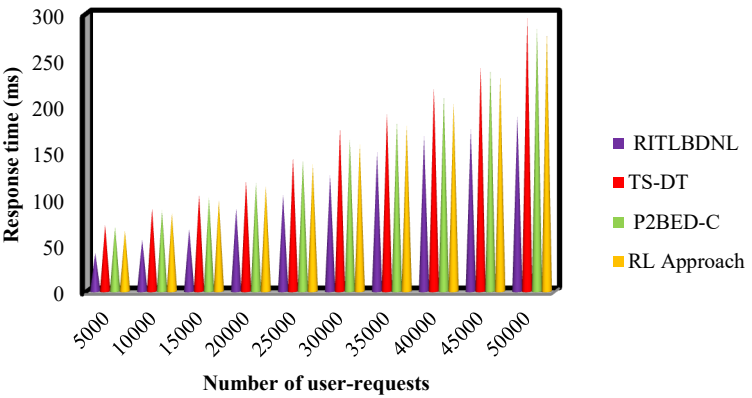

Fig. 8. Analysis of response time

Figure 8 illustrates experimental outcomes of response time with the number of user requests, ranging from 5000 to 50000. As the number of requests enhances, the response time for every method in addition increases. However, the proposed RITLBDNL technique achieves a lower response time compared to existing methods. For instance, with 5000 user requests, the response time for RITLBDNL was observed to be $41.5\,ms$, while [1], [2], and [3] exhibited response times of $72\,ms$, $70\,ms$, and $66\,ms$, respectively. The overall performance results of RITLBDNL are then compared to existing methods, revealing that RITLBDNL minimizes response time consumption by 31%, 28%, and 26% when compared to [1], [2], and [3], respectively. This improvement is achieved using throttled load balancing in the RITLBDNL technique, which effectively performs task migration from overloaded to less loaded virtual machines. Consequently, RITLBDNL minimizes both waiting and processing times for user requests.

5. Discussion. This study compares the proposed RITLBDNL and existing TS-DT [1], P2BED-C [2], and RL Approach [3] based on various parameters, such as load balancing efficiency, throughput, makespan, and response time. The main drawbacks of existing methods such as failure to obtain energy-aware load balancing with a tiny makespan and higher throughput and the failure to employ a resource-aware scheduling approach to assign tasks on VMs. Contrary to existing, Deep Neural Learning and Throttled Load Balancing are utilized in RITLBDNL. By applying this algorithm, the resource status of a virtual machine is examined to find a less

loaded virtual machine. By observing the above table, the results of load balancing efficiency using the proposed RITLBDNL is highly increased than the other existing methods. Also, the response time and makespan of RITLBDNL are greatly reduced than the other works.

Table 1. Comparison table of proposed and existing methods

METHOD	RITLBDNL Technique	TS-DT [1]	P2BED-C [2]	RL Approach [3]
Contribution	To improve load balancing performance using deep neural learning	To allocate tasks using TS-DT	P2BED-C was utilized for data centers	To optimize cloud resource utilization using RL Approach
Merits	Improved throughput and reduced the makespan	Minimized the makespan	Decreased energy consumption	Diminished response time
Demerits	False-positive rate and memory consumption were considered	Energy-aware load balancing was not obtained	Efficiency of the method was not enhanced	Makespan was not efficiently reduced
Load balancing efficiency (%)	98.55	91.5	92.38	93.57
Throughput (requests/sec)	1619.6	1057.9	1113.5	1176.2
Makespan (ms)	46	81	76	70
Response time (ms)	117	165.85	160.7	155.80

Table 1 illustrates a comparison of the proposed RITLBDNL technique and existing TS-DT [1], P2BED-C [2], and RL Approach [3] by using different metrics. Among the three methods, the proposed PCR-AMESCSRO technique provides better performance. The load balancing efficiency was improved by 98.55% using RITLBDNL upon comparison with the three other existing methods. Also, the response time and makespan of the proposed RITLBDNL are obtained as 46 ms and 117 ms which is smaller than the other methods.

6. Conclusion. Balancing the workload is the most important problem in the cloud owing to its dynamic nature. This study introduces a RITLBDNL technique which has been developed to tackle the issue of minimizing makespan and enhancing optimal resource effective load balancing in the cloud. By utilizing the Ruzicka Similarity Index, the

cloud's LB determines the virtual machine resource status for detecting overloaded, less loaded as well as balanced loads. LB performs to dispense workload from heavily loaded virtual machines to minimum loaded ones with higher efficiency. The experimental results also prove that the proposed model has reduced makespan, as well as response time, improved throughput, and efficiency. Compared with various state-of-the-art models, the proposed technique is more efficient. The outcomes of this study have important implications for business applications (i.e., Amazon cloud) to find and classify the resource-efficient VM to allocate the tasks. The less loaded machine needs minimum time and makespan to complete the user requests. Overall, this study provides a valuable contribution to the field of load balancing using DL, and its proposed technique can be extended to other domains where novel DL and optimization are used.

References

1. Mahmoud H., Thabet M., Khafagy M., Omara F. Multiobjective task scheduling in cloud environment using decision tree algorithm. IEEE Access. 2022. vol. 10. pp. 36140–36151.
2. Kumar K. P2BED-C: a novel peer to peer load balancing and energy efficient technique for data-centers over cloud. Wireless Personal Communications. 2022. vol. 123(1). pp. 311–324.
3. Lahande P., Kaveri P., Saini J., Kotecha K., Alfarhood S. Reinforcement Learning approach for optimizing Cloud Resource Utilization with Load Balancing. IEEE Access. 2023. vol. 11. pp. 127567–127577.
4. Nabi S., Ibrahim M., Jimenez J. DRALBA: Dynamic and resource aware load balanced scheduling approach for cloud computing. IEEE Access. 2021. vol. 9. pp. 61283–61297.
5. Sohani M., Jain S. A predictive priority-based dynamic resource provisioning scheme with load balancing in heterogeneous cloud computing. IEEE access. 2021. vol. 9. pp. 62653–62664.
6. Kaviarasan R., Balamurugan G., Kalaiyarasan R. Effective load balancing approach in cloud computing using Inspired Lion Optimization Algorithm. e-Prime-Advances in Electrical Engineering, Electronics and Energy. 2023. vol. 6. DOI: 10.1016/j.prime.2023.100326.
7. Adil M., Nabi S., Aleem M., Diaz V., Lin J. CA-MLBS: content-aware machine learning based load balancing scheduler in the cloud environment. Expert Systems. 2023. vol. 40(4). DOI: 10.1111/exsy.13150.
8. Muthusamy A., Dhanaraj R. Dynamic Q-Learning-Based Optimized Load Balancing Technique in Cloud. Mobile Information Systems. 2023. vol. 2023(1). DOI: 10.1155/2023/7250267.
9. Kruekaew B., Kimpan W. Multi-objective task scheduling optimization for load balancing in cloud computing environment using hybrid artificial bee colony algorithm with reinforcement learning. IEEE Access. 2022. vol. 10. pp. 17803–17818.
10. Mapetu J., Kong L., Chen Z. A dynamic VM consolidation approach based on load balancing using Pearson correlation in cloud computing. The Journal of Supercomputing. 2021. vol. 77(6). pp. 5840–5881.

11. Saroit I., Tarek D. LBCC-Hung: A load balancing protocol for cloud computing based on Hungarian method. Egyptian Informatics Journal. 2023. vol. 24(3). DOI: 10.1016/j.eij.2023.100387.
12. Ashawa M., Douglas O., Osamor J., Jackie R. Retracted Article: Improving cloud efficiency through optimized resource allocation technique for load balancing using LSTM machine learning algorithm. Journal of Cloud Computing. 2022. vol. 11(1). DOI: 10.1186/s13677-022-00362-x.
13. Annie Poornima Princess G., Radhamani A. A hybrid meta-heuristic for optimal load balancing in cloud computing. Journal of grid computing. 2021. vol. 19(2). DOI: 10.1007/s10723-021-09560-4.
14. Mekonnen D., Megersa A., Sharma R., Sharma D. Designing a Component-Based Throttled Load Balancing Algorithm for Cloud Data Centers. Mathematical Problems in Engineering. 2022. vol. 2022(1). DOI: 10.1155/2022/4640443.
15. Haidri R., Alam M., Shahid M., Prakash S., Sajid M. A deadline aware load balancing strategy for cloud computing. Concurrency and Computation: Practice and Experience. 2022. vol. 34(1). DOI: 10.1002/cpe.6496.
16. Pradhan A., Bisoy S., Sain M. Action-Based Load Balancing Technique in Cloud Network Using Actor-Critic-Swarm Optimization. Wireless Communications and Mobile Computing. 2022. vol. 2022(1). DOI: 10.1155/2022/6456242.
17. Udayasankaran P., Thangaraj S. Energy efficient resource utilization and load balancing in virtual machines using prediction algorithms. International Journal of Cognitive Computing in Engineering. 2023. vol. 4. pp. 127–134.
18. Velpula P., Pamula R. EBGO: an optimal load balancing algorithm, a solution for existing tribulation to balance the load efficiently on cloud servers. Multimedia Tools and Applications. 2022. vol. 81(24). pp. 34653–34675.
19. Johora F., Ahmed I., Shajal M., Chowdhory R. A load balancing strategy for reducing data loss risk on cloud using remodified throttled algorithm. International Journal of Electrical and Computer Engineering. 2022. vol. 12(3). pp. 3217–3225. DOI: 10.11591/ijece.v12i3.
20. Khalil M., Shah S., Taj A., Shiraz M., Alamri B., Murawwat S., Hafeez G. Renewable-aware geographical load balancing using option pricing for energy cost minimization in data centers. Processes. 2022. vol. 10(10). DOI: 10.3390/pr10101983.
21. Rajashekar K., Channakrishnaraju Gowda P., Jayachandra A. SCEHO-IPSO: A Nature-Inspired Meta Heuristic Optimization for Task-Scheduling Policy in Cloud Computing. Applied Sciences. 2023. vol. 13(19). DOI: 10.3390/app131910850.
22. Rani P., Singh P., Verma S., Ali N., Shukla P., Alhassan M. An implementation of modified blowfish technique with honey bee behavior optimization for load balancing in cloud system environment. Wireless Communications and Mobile Computing. 2022. vol. 2022. DOI: 10.1155/2022/3365392.
23. Hung L., Wu C., Tsai C., Huang H. Migration-based load balance of virtual machine servers in cloud computing by load prediction using genetic-based methods. IEEE Access. 2021. vol. 9. pp. 49760–49773.
24. Devi K., Sumathi D., Vignesh V., Anilkumar C., Kataraki K., Balakrishnan S. CLOUD load balancing for storing the internet of things using deep load balancer with enhanced security. Measurement: Sensors. 2023. vol. 28. DOI: 10.1016/j.measen.2023.100818.
25. Adaikalaraj J., Chandrasekar C. To improve the performance on disk load balancing in a cloud environment using improved Lion optimization with min-max algorithm. Measurement: Sensors. 2023. vol. 27. DOI: 10.1016/j.measen.2023.100834.

Ellakkiya M. — Research scholar, Department of computer science, Cauvery College For Women (Autonomous); Research scholar, Thanthai Periyar Government Arts and Science

College (Autonomous), Affiliated to Bharathidasan University. Research interests: cloud computing, machine learning. The number of publications — 15. ellakkiya.researchscholar@gmail.com; 620023, Tiruchirappalli, India; office phone: +9994683100.

Ravi T.N. — Associate professor of pg & research, Department of computer science, Jamal Mohamed College (Autonomous), Tiruchirappalli, Affiliated to Bharathidasan University. Research interests: parallel processing, network computing, genetic algorithms, artificial intelligence, data mining. The number of publications — 73. proftnravi@gmail.com; 36/2, Race Course Road, Khajamalai, 620023, Tiruchirappalli, India; office phone: +91(0431)233-1135.

Panneer Arokiaraj S. — Associate professor, Department of computer science, Thanthai Periyar Government Arts and Science College (Autonomous), Affiliated to Bharathidasan University. Research interests: data compression, biometric authentication, data mining. The number of publications — 15. drpancs@gmail.com; 36/2, Race Course Road, Khajamalai, 620023, Tiruchirappalli, India; office phone: +91(0431)242-0079.

УДК 004 **DOI** 10.15622/ia.23.6.9

М. Эллаккия, Т. Рави, С. Паннир Арокиарадж

ИНДЕКСНОЕ РЕГУЛИРУЕМОЕ ГЛУБОКОЕ НЕЙРОННОЕ ОБУЧЕНИЕ РУЖИЧКИ ДЛЯ РЕСУРСОЭФФЕКТИВНОЙ БАЛАНСИРОВКИ НАГРУЗКИ В ОБЛАЧНОЙ СРЕДЕ

Эллаккия М., Рави Т., Паннир Арокиарадж С. **Индексное регулируемое глубокое нейронное обучение Ружички для ресурсоэффективной балансировки нагрузки в облачной среде.**

Аннотация. Облачные вычисления (CC) являются известной технологией, которая позволяет пользователям и организациям получать доступ к сервисам в соответствии с их требованиями. Этот метод вычислений предлагает хранилище, платформы развертывания и подходящий доступ к веб-сервисам через интернет. Балансировка нагрузки является важным фактором оптимизации вычислительных ресурсов и хранения. Она направлена на разумное распределение рабочей нагрузки между каждой виртуальной машиной. Было разработано несколько традиционных методов балансировки нагрузки, которые доступны в литературе. Однако достижение эффективной балансировки нагрузки с минимальным временем завершения и улучшенной пропускной способностью остается сложной задачей. Для повышения эффективности балансировки нагрузки был разработан новый метод, известный как индексированный регулируемый метод Ружички балансировки нагрузки глубокого нейронного обучения (RITLBDNL). Основная цель RITLBDNL состоит в том, чтобы повысить пропускную способность и минимизировать время выполнения работы в облаке. В методе RITLBDNL модель глубокого нейронного анализа включает входной слой, два скрытых слоя и выходной слой для улучшения производительности балансировки нагрузки. На входном слое собираются задачи пользователей облака и отправляются на скрытый слой 1. На этом слое балансировщик нагрузки в облачном сервере анализирует состояние ресурсов виртуальной машины в зависимости от энергии, пропускной способности, объема памяти и ЦПУ с использованием индекса сходства Ружички. Затем виртуальные машины классифицируются как перегруженные, слабо загруженные или сбалансированные. Результаты анализа передаются на скрытый слой 2, где выполняется регулируемая балансировка нагрузки для распределения нагрузки с сильно загруженных виртуальных машин на минимально загруженные. Облачный сервер эффективно распределяет рабочую нагрузку между виртуальными машинами с более высокой пропускной способностью и меньшим временем отклика для обработки огромного количества входящих задач. Для оценки результатов экспериментов предложенный метод сравнивается с другими существующими методами балансировки нагрузки. Результат показывает, что предложенный метод RITLBDNL обеспечивает эффективность балансировки нагрузки с увеличением на 7%, пропускной способностью на 46%, уменьшением времени завершения на 41% и времени отклика на 28% по сравнению с традиционными методами.

Ключевые слова: облачные вычисления, балансировка нагрузки, глубокое обучение, индекс сходства Ружички, регулируемая балансировка нагрузки.

Литература

1. Mahmoud H., Thabet M., Khafagy M., Omara F. Multiobjective task scheduling in cloud environment using decision tree algorithm. IEEE Access. 2022. vol. 10. pp. 36140–36151.

2. Kumar K. P2BED-C: a novel peer to peer load balancing and energy efficient technique for data-centers over cloud. Wireless Personal Communications. 2022. vol. 123(1). pp. 311–324.
3. Lahande P., Kaveri P., Saini J., Kotecha K., Alfarhood S. Reinforcement Learning approach for optimizing Cloud Resource Utilization with Load Balancing. IEEE Access. 2023. vol. 11. pp. 127567–127577.
4. Nabi S., Ibrahim M., Jimenez J. DRALBA: Dynamic and resource aware load balanced scheduling approach for cloud computing. IEEE Access. 2021. vol. 9. pp. 61283–61297.
5. Sohani M., Jain S. A predictive priority-based dynamic resource provisioning scheme with load balancing in heterogeneous cloud computing. IEEE access. 2021. vol. 9. pp. 62653–62664.
6. Kaviarasan R., Balamurugan G., Kalaiyarasan R. Effective load balancing approach in cloud computing using Inspired Lion Optimization Algorithm. e-Prime-Advances in Electrical Engineering, Electronics and Energy. 2023. vol. 6. DOI: 10.1016/j.prime.2023.100326.
7. Adil M., Nabi S., Aleem M., Diaz V., Lin J. CA-MLBS: content-aware machine learning based load balancing scheduler in the cloud environment. Expert Systems. 2023. vol. 40(4). DOI: 10.1111/exsy.13150.
8. Muthusamy A., Dhanaraj R. Dynamic Q-Learning-Based Optimized Load Balancing Technique in Cloud. Mobile Information Systems. 2023. vol. 2023(1). DOI: 10.1155/2023/7250267.
9. Kruekaew B., Kimpan W. Multi-objective task scheduling optimization for load balancing in cloud computing environment using hybrid artificial bee colony algorithm with reinforcement learning. IEEE Access. 2022. vol. 10. pp. 17803–17818.
10. Mapetu J., Kong L., Chen Z. A dynamic VM consolidation approach based on load balancing using Pearson correlation in cloud computing. The Journal of Supercomputing. 2021. vol. 77(6). pp. 5840–5881.
11. Saroit I., Tarek D. LBCC-Hung: A load balancing protocol for cloud computing based on Hungarian method. Egyptian Informatics Journal. 2023. vol. 24(3). DOI: 10.1016/j.eij.2023.100387.
12. Ashawa M., Douglas O., Osamor J., Jackie R. Retracted Article: Improving cloud efficiency through optimized resource allocation technique for load balancing using LSTM machine learning algorithm. Journal of Cloud Computing. 2022. vol. 11(1). DOI: 10.1186/s13677-022-00362-x.
13. Annie Poornima Princess G., Radhamani A. A hybrid meta-heuristic for optimal load balancing in cloud computing. Journal of grid computing. 2021. vol. 19(2). DOI: 10.1007/s10723-021-09560-4.
14. Mekonnen D., Megersa A., Sharma R., Sharma D. Designing a Component-Based Throttled Load Balancing Algorithm for Cloud Data Centers. Mathematical Problems in Engineering. 2022. vol. 2022(1). DOI: 10.1155/2022/4640443.
15. Haidri R., Alam M., Shahid M., Prakash S., Sajid M. A deadline aware load balancing strategy for cloud computing. Concurrency and Computation: Practice and Experience. 2022. vol. 34(1). DOI: 10.1002/cpe.6496.
16. Pradhan A., Bisoy S., Sain M. Action-Based Load Balancing Technique in Cloud Network Using Actor-Critic-Swarm Optimization. Wireless Communications and Mobile Computing. 2022. vol. 2022(1). DOI: 10.1155/2022/6456242.
17. Udayasankaran P., Thangaraj S. Energy efficient resource utilization and load balancing in virtual machines using prediction algorithms. International Journal of Cognitive Computing in Engineering. 2023. vol. 4. pp. 127–134.

18. Velpula P., Pamula R. EBGO: an optimal load balancing algorithm, a solution for existing tribulation to balance the load efficiently on cloud servers. Multimedia Tools and Applications. 2022. vol. 81(24). pp. 34653–34675.
19. Johora F., Ahmed I., Shajal M., Chowdhory R. A load balancing strategy for reducing data loss risk on cloud using remodified throttled algorithm. International Journal of Electrical and Computer Engineering. 2022. vol. 12(3). pp. 3217–3225. DOI: 10.11591/ijece.v12i3.
20. Khalil M., Shah S., Taj A., Shiraz M., Alamri B., Murawwat S., Hafeez G. Renewable-aware geographical load balancing using option pricing for energy cost minimization in data centers. Processes. 2022. vol. 10(10). DOI: 10.3390/pr10101983.
21. Rajashekar K., Channakrishnaraju Gowda P., Jayachandra A. SCEHO-IPSO: A Nature-Inspired Meta Heuristic Optimization for Task-Scheduling Policy in Cloud Computing. Applied Sciences. 2023. vol. 13(19). DOI: 10.3390/app131910850.
22. Rani P., Singh P., Verma S., Ali N., Shukla P., Alhassan M. An implementation of modified blowfish technique with honey bee behavior optimization for load balancing in cloud system environment. Wireless Communications and Mobile Computing. 2022. vol. 2022. DOI: 10.1155/2022/3365392.
23. Hung L., Wu C., Tsai C., Huang H. Migration-based load balance of virtual machine servers in cloud computing by load prediction using genetic-based methods. IEEE Access. 2021. vol. 9. pp. 49760–49773.
24. Devi K., Sumathi D., Vignesh V., Anilkumar C., Kataraki K., Balakrishnan S. CLOUD load balancing for storing the internet of things using deep load balancer with enhanced security. Measurement: Sensors. 2023. vol. 28. DOI: 10.1016/j.measen.2023.100818.
25. Adaikalaraj J., Chandrasekar C. To improve the performance on disk load balancing in a cloud environment using improved Lion optimization with min-max algorithm. Measurement: Sensors. 2023. vol. 27. DOI: 10.1016/j.measen.2023.100834.

Эллаккия М. — научный сотрудник, факультет компьютерных наук, Колледж Кавери для женщин (автономный); научный сотрудник, Государственный колледж искусств и наук имени Тхантхай Перияра (автономный), филиал Университета Бхаратхидасан. Область научных интересов: облачные вычисления, машинное обучение. Число научных публикаций — 15. ellakkiya.researchscholar@gmail.com; 620023, Тируччираппалли, Индия; р.т.: +9994683100.

Рави Т.Н. — доцент кафедры, факультет компьютерных наук, Колледж Джамаля Мохаммеда (автономный), Тируччираппалли, филиал Университета Бхаратхидасан. Область научных интересов: параллельная обработка, сетевые вычисления, генетические алгоритмы, искусственный интеллект и интеллектуальный анализ данных. Число научных публикаций — 73. proftnravi@gmail.com; Ипподром, Хаджамалай, 36/2, 620023, Тируччираппалли, Индия; р.т.: +91(0431)233-1135.

Паннир Арокиарадж С. — доцент кафедры, кафедра компьютерных наук, Государственный колледж искусств и науки Тантай Перияр (автономный), Университет Бхаратидасан. Область научных интересов: сжатие данных, биометрическая аутентификация, интеллектуальный анализ данных. Число научных публикаций — 15. drpancs@gmail.com; Ипподром, Хаджамалай, 36/2, 620023, Тируччираппалли, Индия; р.т.: +91(0431)242-0079.