*Computers, Materials & Continua*
DOI: 10.32604/cmc.2022.028331
***Article***

Tech Science Press

# P-ACOHONEYBEE: A Novel Load Balancer for Cloud Computing Using Mathematical Approach

**Sunday Adeola Ajagbe[1], Mayowa O. Oyediran[2], Anand Nayyar[3,*], Jinmisayo A. Awokola[4] and Jehad F. Al-Amri[5]**

[1]Computer Engineering Department, Ladoke Akintola University of Technology, LAUTECH, Ogbomoso, Nigeria
[2]Computer Science Department, Ajayi Crowther University, Oyo, Nigeria
[3]Graduate School, Faculty of Information Technology, Duy Tan University, Da Nang, Vietnam
[4]Department of Computer Sciences, Precious Cornerstone University, Ibadan, Nigeria
[5]Department of Information Technology, College of Computers and Information Technology, Taif University, Taif, 21944, Saudi Arabia
*Corresponding Author: Anand Nayyar. Email: anandnayyar@duytan.edu.vn
Received: 07 February 2022; Accepted: 12 April 2022

**Abstract:** Cloud computing is a collection of disparate resources or services, a web of massive infrastructures, which is aimed at achieving maximum utilization with higher availability at a minimized cost. One of the most attractive applications for cloud computing is the concept of distributed information processing. Security, privacy, energy saving, reliability and load balancing are the major challenges facing cloud computing and most information technology innovations. Load balancing is the process of redistributing workload among all nodes in a network; to improve resource utilization and job response time, while avoiding overloading some nodes when other nodes are underloaded or idle is a major challenge. Thus, this research aims to design a novel load balancing systems in a cloud computing environment. The research is based on the modification of the existing approaches, namely; particle swarm optimization (PSO), honeybee, and ant colony optimization (ACO) with mathematical expression to form a novel approach called P-ACOHONEYBEE. The experiments were conducted on response time and throughput. The results of the response time of honeybee, PSO, SASOS, round-robin, PSO-ACO, and P-ACOHONEYBEE are: 2791, 2780, 2784, 2767, 2727, and 2599 (ms) respectively. The outcome of throughput of honeybee, PSO, SASOS, round-robin, PSO-ACO, and P-ACOHONEYBEE are: 7451, 7425, 7398, 7357, 7387 and 7482 (bps) respectively. It is observed that P-ACOHONEYBEE approach produces the lowest response time, high throughput and overall improved performance for the 10 nodes. The research has helped in managing the imbalance drawback by maximizing throughput, and reducing response time with scalability and reliability.

**Keywords:** ACO; cloud computing; load balancing; swarm intelligence; PSO; P-ACOHONEYBE; honeybee swarm

## 1 Introduction

The requirement to allow an increasing degree of flexibility, adaptability and autonomy in a cloud environment has influenced the development of distributed systems [1]. Cloud computing, similar to public utility, is a form of internet-based computing that makes uses shared resources, software, and data, which are made available to computers and other devices on-demand [2,3]. It is made up by aggregating two terms in the field of technology, "cloud" and "computing". In the realm of technology, a cloud is a collection of diverse resources, a web of massive infrastructure, and the phrase "Cloud" has no meaning. Computation in the cloud is conducted and the goal is to maximize resource utilization while maintaining high availability at a low cost [4]. One of the most attractive applications for cloud computing is the concept of distributed information processing [5]. This is particularly clear in the scenario of comparing cloud, grid, and cluster computing that cloud is not the combination of clusters and grid, but are next generation to them [6,7]. In cloud computing, resource sharing is an on-demand service and the particle swarm optimization (PSO) method is among the promising tools [8]. As a result, cloud computing is a scenario in which a user may have safe access to any type of infrastructure, software, or platform at a reduced cost on demand in an easy-to-use manner [9,10].

Cloud computing has some characteristics that include the following: Virtualization, Availability and Scalability of Cloud Computing are the distinct attributes that differentiate it from Cluster and Grid Computing [11–15]. Cloud computing uses scalability guidelines to control resource requirements, allowing businesses to save a significant amount of money on IT infrastructure. In cloud computing environments, hosting providers are unable to have full control over the running application because it is usually under the supervision of the company that subscribes to the system that has produced the application [16–18]. This makes it difficult for the hosting providers to monitor the application behavior and prepare the systems for necessary actions to improve the overall system stability and availability [19–23]. The aforementioned fact leads to uneven distribution of network traffic coming into the computer system on a network side, as well as on a server-side [24–26]. Despite cloud computing's bright future, many crucial issues must be addressed before they can be fully realized. One of the most attractive applications for cloud computing is the concept of distributed information processing. Security, privacy, energy saving, reliability and load balancing are the major challenges facing cloud computing and most information technology innovations. One of the major challenges in the cloud computing environment is load balancing and these challenges are not unconnected with the response time and throughput [27–29]. High response time and low throughput are common difficulties in load balancing, and it affects sharing the limited resources in a cloud environment. Therefore, this research seeks to develop an improved load balancing system to optimize resources used in the cloud computing environment. The specific objectives of this research are to:

i) conduct extensive literature studies about state-of-the-art research on load balancing.
ii) design a novel algorithm for load balancing system in a cloud computing environment using modify conventional PSO, ACO, and Honeybee mathematical equations.
iii) experimentation of self-partitioning and clustering of resources and also serve as balancer within the cloud environment.
iv) evaluate the performance of the developed load balancing strategy using response time, and throughput as metrics.

### Organization of the paper

The paper is organized as: Section 2 discusses related works on load balancing in a cloud computing environment, section 3 elaborates the methodology adopted for the research, Section 4

presents the implementation, results, and analysis. Finally, Section 5 concludes the paper with future scope.

## 2 Review of Related Works

Topcuoglu, et al., (2020) [28] described load balancing as the process of redistributing a distributed system's general workload among all nodes (disk drivers, network connectivity, and CPU) to enhance resource utilization, job response time and overall efficiency, while preventing situations where some nodes are overloaded but some nodes are underloaded. Choudhari, *et al*., (2018) [29] opined that a few key goals of load balancing mechanisms have been addressed in various studies. These goals include reducing the time it takes to respond to a job while maintaining acceptable delays, keeping the system stable, having the ability to tolerate mistakes, and improving and maintaining cloud system availability among others.

One of the limitations of achieving optimum efficiency in the networks is imbalance drawback. Random allocation and round-robin algorithms were proposed by Hoang et al. [30] to address the imbalance drawback in the load balancing system, the algorithms were evaluated using response time and the results were 9103 and 9134 (ms) for the random allocation and round-robin algorithms respectively. The problem of reassigning total loads to individual nodes of the collective system to achieve the lower response time and resource usage load balancing system was attempted by li et al. [31], the study used PSO and evaluated using response time and migration time. The study reported 8991 and 4899 (ms) for response time and migration time respectively. Ant Colony Optimization (ACO) was used to ensure load balancing and safe packet delivery by Kabirzadeh et al. [32], wherein the research used response time and migration time as performance evaluation metrics and results were 9001 and 6128 (ms) for response time and migration time respectively. The study concluded that these results may not guarantee the regulation of traffic and workload.

An improved system was developed by Kabirzadeh et al. [32], in which the objectives were to avert system failure by regulating input traffic and stopping the workload from being sent to resources that have become overloaded and unresponsive. The overall aim was to achieve a frequent load balancing system. The improved algorithm in the study was based on the ACO and Honeybee mathematical expression to develop ACOBee. The two existing algorithms were compared alongside the newly developed model using response time as a metric. The study reported 8991, 8919 and 7901 (ms) response times for ACO, Honeybee, and ACOBee respectively. In a bid to reduce complexity in the cloud computing environment and ensure keeping the system stable, having the ability to tolerate mistakes according to [33]. Honeybee and round-robin algorithms were used in the design and implementation of a stable system in the cloud environment [34]. Kiran, et al., (2021) [35] presented a better public cloud load balancing architecture based on the partitioning of the cloud with a switch mechanism to select various systems for various scenarios. The technique used game theory to improve the efficiency of the load balancing method on the public cloud. The model developed was compared with other ones and it was discovered that the system was able to partition a public cloud and also can switch between two different strategies but the system suffers from optimized resource utilization.

Qi, et al., 2019 [36] indicated a partitioning and load balancing method for a cloud environment. The mechanism for partitioning a cloud is revealed in the paper, as well as a comparison of several algorithms for balancing dynamic demand. The comparison of the Ant Colony and Honey Bee algorithms yielded the conclusion that the Honey Bee method is the best in typical load conditions, while the basic round-robin algorithm was used when the partitions were idle. In the results, Ant Colony outperforms HoneyBee when compared. However, the algorithm suffers from non-dynamic

partitioning of the resource. Because the academics approached it from various angles, perspectives, technologies, and backgrounds, different definitions for load balancing in cloud computing have been created to date. cloud computing has a number of benefits and challenges. Benefits of cloud computing include but are not limited to scalability, elasticity, mobility, low infrastructures, increased data storage, availability, billing and payment [37]. Considering the challenges, which include; security and privacy, interoperability, energy-saving, disaster recovery and load balancing [38]. Load balancing, which was highlighted as a key feature of cloud computing and is still considered an open study topic and has a significant effect on determining resource availability [39]. Lakra et al., (2015) [40] remarked that the feature inherited from grid computing has been challenged by scalability and post a threat to load balancing. Thus, the PSO algorithm was used to address and the performance was measured using response time and throughput. The response time of 8100 (ms) and throughput of 4901 (bps) were reported.

Cardellini et al., (2015) [41] remarked that load balancing algorithms should be stable enough to work with a wide range of applications. An attempt was made towards ensuring a stable load balancer for a wide range of applications, the following checklists are recommended for designing more efficient load balancing algorithms based on [42]. The research reported improvement in complexity, scalability, and fault tolerance performance. In terms of load management, independent and dependent scheduling approaches are considered to be the most important strategy. Dependent scheduling is gaining popularity [43]. Dependent scheduling is appropriate for jobs that have dependent organized patterns. Usually, each job is made up of a number of interconnected tasks. Hence, task execution is dependent on one another through the limited improvement was achieved in [44]. Failures in the execution of dependent tasks, as opposed to failures in the execution of independent tasks, have an impact on the overall system performance. At the moment, no viable algorithm exists for load balancing on dependent structured jobs.

Previous load balancing solutions were software-based and time-based, with optimization done locally. For load balancing, a clever solution is proposed by He et al. [45]. The suggested approach performs global optimization over time. Load balancing was based on hardware in the proposed method, and load balancing is computed at two levels: the entire system and the virtual machine or server. The proposed strategy improves resource use. Mtshali et al., 2019 [46] suggested a method for scheduling based on virtualization technology for energy consumption optimization and real-time delay in fog computing technology. Four scheduling task policies were implemented and simulations were conducted with iFogSimtooland. Improvements of 11% energy consumption, 7.78% average task delay, 4.4% network usage, and 15.1% execution time were achieved in the study. Results show that FCFS was better than the existing methods.

In a cloud computing study, Somwang (2020) [47] proposed an effective load balance control solution based on HA Proxy, the purpose is to receive and distribute workload to the computer server so that processing resources can be shared. The study used a round-robin scheduling technique to manage the cloud storage systems' resources efficiently, with a focus on effective workload balancing and a dynamic replication mechanism. Results revealed that the proposed technique may enhance load balancing performance in cloud computing by 1,000 requests /6.31 seconds while also reducing false alarms. Sliwko (2019) [48] work focused on the development of a mechanism for dynamic allocation jobs without overloading cloud nodes, resulting in system stability at a low cost. A new taxonomy and classification system for three types of schedulers, namely, OS-level, Cluster, and Big Data, highlights their distinct evolution and goals. Extensive trials on the University of Westminster HPC cluster were conducted for the project, and the positive results are presented together with lengthy comments and a conclusion. Finally, with a stable simulation environment in place, the project was able to go forward

with the development of load balancing solutions. However, the study did not report the metrics used to evaluate the proposed approach.

For the purpose of increasing the efficiency of cloud services, an adaptive load balancing algorithm (ALBA) based on a load prediction technique was presented by Mao et al. (2013) [49]. The ALBA system can reclaim the cluster's resources when the load in the cluster of virtual machines was less than the minimal level. While the load on the virtual machine cluster exceeded the maximum threshold, the ALBA added more virtual machines as needed to balance the computing load and ensure response time. Extensive tests with CloudSim show that the suggested ALBA was enhanced resource usage while also reducing task response time to an extent. Abiodun et al., (2020) [50] offers maximum-minimum round-robin (MMRR) as a new load balancer, the algorithm which combined maximum-minimum and round-robin algorithms to assign work with long execution times to maximum-minimum and tasks with the shortest execution time to round-robin. According to the conclusions of the study, the MMRR has had a substantial impact on cloud services. From Throttled and MMRR, the data center's loading time is good, however, round-robin was the worst. Based on the overall reaction time and cost-effectiveness, MMRR outperformed the other algorithms with an outcome of 89%. The study concluded that improving customer happiness in cloud services should be implemented.

Arising from the review of related works, a need for efficient task scheduling or load balancing algorithms for optimum resource utilization, response time evade of application unresponsiveness and non-dynamic partitioning of resources in the cloud computing environment was observed. Load balancing which is the process of redistributing workload among all nodes in a network, to improve resource utilization and task response time while preventing overloaded some nodes and others are underloaded or idle is a major challenge. In addition, there are difficulties in sharing the limited resources in a cloud environment, and this may reduce the productivity and efficiency of the system. The importance of load balancing in cloud computing and elastic scalability cannot be overstated. As cloud computing is evolving in the computing field, one of its attractive applications is the concept of distributed information processing. According to the reviewed literature, the innovation in distributed information processing of the cloud computing environment is threatened by security, privacy, energy-saving, and load balancing. Load balancing is the process of redistributing a distributed system's general workload among all nodes to enhance the utilization of resources and job response time, while preventing situations where some nodes are overloaded and others are idle or underloaded. Several attempts have been made in the past but with a minimal solution to the problem. A number of studies used PSO, ACO, HoneyBee and a few others such as Firefly. The response time, throughput, waiting time and migration time are common metrics to measure the performance evaluation used by most of these studies in cloud computing. Hence, this research was informed, by developing a novel algorithm for an efficient and effective distribution of load across the cloud computing system, because high response time and low throughput were observed as main gaps of load balancing in the reviewed works.

## 3 Proposed Methodology

In developing a novel load balancing system for a cloud computing environment in this research, the following research approaches were used:

Modification of mathematical expressions for the developed load balancing strategy using PSO, ACO, and HoneyBee mathematical expression to perform self–partitioning of resources and also to serve as a load balancer in the cloud computing environment

i) Development of an algorithm for the load strategy: The design of the algorithm was done sequentially, wherein the output of PSO serves as input for the ACO and HoneyBee which acted as the load balancer.
ii) Implementation of the developed algorithm: The developed algorithm was implemented in Microsoft Azure cloud-integrated environment, which is an open and flexible cloud environment that allows developers to build, deploy and manage applications in any programming language.
iii) Verification of the algorithm for Quality of Service (QoS): The designed algorithm was verified for QoS to ascertain its correctness.
iv) Evaluation of the algorithm: The developed system was tested with other existing techniques like: honeybee, PSO, round-robin, SASOS and PSO-ACO to evaluate the performance of the developed system using; response time, and throughput as metrics.

### *3.1 System Model and Mathematical Expressions for the Development of a Load Balancing System*

The optimization within the framework of an itinerary is an NP-complete problem that entails self-partition and clustering of resources of the cloud environment with the set objective to evade/reduce the overloading of components in cloud computing. The self-partitioning and clustering unit of the system was developed using Eqs. (3)–(8). In the traditional PSO according to Shishira et al. [20], the inertia weight ($\infty$) which is the determining parameter that helps in the search for global best of the particle cannot commerce due to the fact that the velocity update is trapped at the highest possible value because $\omega = +\infty$, the particle velocity is at its maximum and skips multiple optimal solutions. The parameter $\omega$ (inertia weight) was modified by finding a finite value for inertia weight $\omega$ as presented in Eq. (5) to prevent the premature convergence of the particles towards achieving a maximum global best result. The equation for updating the velocity of the particles and particle position is presented in Eqs. (3) and (4) with a source from [15] who proved Eqs. (1) and (2) respectively. The position of the particle is changed by adding a velocity, $v_i(t)$, to the current position presented in Eq. (1) which meant the following:

$$\text{xi}\ (\text{t} + 1) = \ \text{xi}(\text{t}) + \ \text{vi}(\text{t} + 1) \tag{1}$$

$$with\ xi(0) \sim U(xmin, xmax).$$

$$Vi + 1 = \infty Vi + c1rand1 * (pbest - xi) + c2rand2 * (gbest - xi) \tag{2}$$

$$v_{i(t+1)} = \omega v_{i(t)} + c_1 \left(P_{i(t)} - x_{i(t)}\right) + c_2 \left(g_{(t)} - x_{i(t)}\right) \tag{3}$$

$$x_{i(t+1)} = x_{i(t)} + v_{i(t+1)} \tag{4}$$

Where in,

$c_1$, *and* $c_2$ are the acceleration coefficient

$\omega$ is the inertia weight

$x_{i(t)}$ is the particle's initial position

$P_{i(t)}$ is the particle personal best position

$g_{(t)}$ is the best position in the entire network on a worldwide scale.

$x_{i(t+1)}$ is the particle new updated position

$v_{i(t)}$ is the particle initial velocity

$v_{i(t+1)}$ is the particle's new velocity

$$\omega_j^{\ t} = \omega_{min} * (\omega_{max} - \omega_{\min}) \tag{5}$$

where $\omega_{\ \min}$ and $\omega_{\ \max}$ are the minimum and maximum inertia weight values, t is the current number of iterations. To determine the fitness value of each node position, the equation is presented in Eq. (6), where $N$ is the swarm size, $f_i$ is the fitness of particle i.

$$f_{i(x)} = \sum_{i=1}^{N} x_i^2 \tag{6}$$

In developing the part that determines the next available node to be loaded with the request, which is based on modifying the ant equation and honeybee equation to determine the busy, idle, and normal node in the cloud environment, the following assumptions were made; wherein a set of n variables exists; where n is the number of nodes on a cloud itinerary. Then, the building graph is fully connected and corresponds to a map of hosts, with each host corresponding to a node and possible edges representing connections between hosts. An ant begins in a random host. The next host *j* for an ant to choose (host j), given it is currently in host *i* and has already visited the hosts in partial solution $s_p$, is as presented in Eq. (8)

$$P\left(j|s_p, i\right) = \frac{\tau_{il}^{\alpha} * \eta(il)^{\beta}}{\sum_{I \in Nsp} \tau_{il}^{\alpha} * \eta(il)^{\beta}}, \forall j \in N(s_p) \tag{7}$$

The relevance of the pheromone and the heuristic function is determined by the weight parameters $\alpha$ and $\beta$. The equation states that only hosts that have not yet been visited are considered, and it constitutes the neighborhood N ($s_p$). For the honeybee Equation in (9);

$$P_i = \frac{fit_i}{\sum_{i=0}^{N} fit_i} \tag{8}$$

$$T_n = \sum_{i=1}^{T_n} P_i \tag{9}$$

where Tn is the total number of the job of the same priority, the Tn value is calculated for each server for different priorities which range from 1 to m, and the node having more number of same priority job as the arrived job is assigned for processing and $p_i$ is the probability of obtaining the *ith* employed bee.

### *3.2 Architecture and Working for Load Balancing Strategy in Cloud Computing.*

In developing an algorithm for load balancer strategy in the cloud environment, the process involved was done sequentially which means, the output of PSO is given as the input to ACO and honeybee algorithms. PSO algorithm is to perform self-partitioning of the cloud computing resources considering resource wastage. ACO algorithm and honeybee algorithm are used as the local and global exploration and exploitation of the explored search space in the developed hybrid algorithm. Fig. 1 presents the flowchart of the developed load balancing strategy, and the developed algorithm is presented mathematically in Algorithm 1. The Flowchart of the developed load balancing strategy has the complete strategy and evaluation of the system.

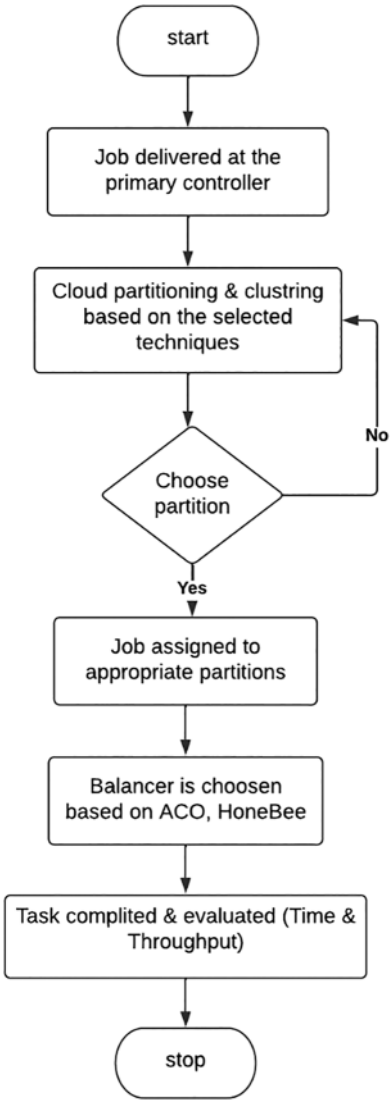


**Figure 1:** Flowchart of the developed load balancing strategy

**Algorithm 1:** The P-ACOHONEYBEE Approach for the load balancing strategy

*Set parameters* ***For*** *each node i do*

*Randomly initialize the position* $(x_i)$ *and velocity* $(v_i)$ *of all particles* $p_i \leftarrow x_i$

*Evaluate the fitness value of each node's position*

$f_{i(x)} = \sum_{i=1}^{N} x_i^2$

*Update inertia weight* $\omega$

$\omega_j^{\ t} = \omega_{min} * (\omega_{max} - \omega_{min})$

***End for***

***While*** *stopping criteria is not met* ***do***

(Continued)

**Algorithm 1:** Continued

*Update the* $p_{best(i)}$
*If* $f(x_i) > p_i$ *then*
$p_i \leftarrow f(x_i)$
*end if*
*Update the* $g_{best(i)}$ *by*
$g_{best} = max(p_{best(i)})$
$g_{best} \leftarrow i$
*Update* $v_i$

$$v_{i(t+1)} = \omega_j{}^t v_{i(t)} + r_1 c_1 \left(p_{i(t)} - x_{i(t)}\right) + r_2 c_2 \left(g_{(t)} - x_{i(t)}\right)$$

*Update* $x_i$
$x_{i(t+1)} = x_{i(t)} + v_{i(t+1)}$
*End*.
*Inputs collection*.
*The setting of the servers with resource capacity*.
*The setting of the VMs with their resource requirements*.
*Initialization*.
*// Particle population size, number of ants, bees and maximum number of iterations MAX_ITER are initialized //*
*// Generate initial population by VM placement solutions based on power consumption and resource wastage of each server //*
*Initiate load Balancer, (Start node [server]) to the first node, obtain data and back to the server*
*(1) Determine Sub-optimal routes from start_node to end_node (termination node)*
*// form numbers of nodes,* $2^m$ *numbers of sub-optimal routes will be obtained*.
*(2) Compute path cost for each of the connected nodes on the sub-optimal route*.
*(3) Calculate the possible total distance for each sub-optimal route*
*//* $2^m$ *numbers of distances will be obtained*
*(4) Value ordering of distance*
*(5) Update KB with RP parameter in the new routing path*
*(6) Update Start_node = current_node*
*End_node = Server*
***Until*** *End_condition*.

### *3.3 Experimental Setup and Simulation Setting*

The experimental setup and simulation setting for the verification of the P-ACOHONEYBEE approach developed for Quality of Service (QoS) was conducted. The analysis method of Microsoft azure cloud was used to verify the developed algorithm for Quality of Service. The verification was done to ascertain the correctness of the implemented strategy with respect to the conceptual system.

***Implementation of Parameter for the Developed Algorithm of Load Balancing Strategy***

The implementation was conducted in a Microsoft azure cloud-integrated environment. Microsoft Azure cloud is an open and flexible cloud environment that allows developers to build, deploy and manage applications in any programming language. It also allows the integration of public cloud applications with any existing information technology environment. The developed load balancing

strategy was executed using a combination of inputs. The input was varied to determine the effect of response time, and throughput, of the developed strategy. The implementation was both interactively and automatically with the following parameters settings according to shihira et al. [20]. The virtual machines (255 GB, 512 GB, and 1000 GB of memory), 2000 tasks, two data centers, Bandwidth (1000 MB each), Traffic Type – UDP, System used - Windows 10 Ultimate 64-bit operating system. An Intel (R) Core ™ i5 CPU with a speed of 2.7 GHz, 4 GB RAM and 500 GB hard disk drive were used. The developed system also used Microsoft Azure cloud-integrated environment on Windows 10 Ultimate 64-bit operating system, Intel (R) Core ™ i5 CPU with a speed of 2.7 GHz, 4 GB RAM and 500 GB hard disk drive, and results obtained were evaluated using response time, and throughput as evaluation metrics.

***Evaluation of the Developed Load Balancing Strategy***

The performance of the developed load balancing strategy was evaluated by comparing it with existing load balancing systems, which are as follows; honeybee, particle swarm optimization (PSO), round-robin, simulated annealing symbiotic organisms search (SASOS), and particle swarm optimization – ant colony optimization (PSO-ACO). The number of nodes used to determine the effect of the approach was 10 nodes noting that this number of nodes is sufficient for the evaluation of the strategies. Response time and throughput time were used as evaluation metrics.

***Response time***

While the total amount of time that has elapsed between the start and completion of a task or request (round-trip time), the response time is the amount of time it takes for the scheduling algorithm to respond to a task. It can be calculated using the following:

$$T_r = \boldsymbol{T}_f - T_A \tag{10}$$

where $\boldsymbol{T}_f$ is the first task completion time;

$\boldsymbol{T}_A$ is time for the first duty to arrive.

***Throughput***

The average data rate at which data or messages are successfully sent over a given communications link is referred to as Throughput, and it is measured in bits per second (bps).

$$\text{Throughput} = \frac{sum\ \ (number\ of\ \ successful\ packets\ ) * (average\ packet\_size)}{total\ time\ spent\ indelivering\ that\ amount\ of\ \ data} \tag{11}$$

## 4 Results and Analysis

The result is a novel P-ACOHONEYBEE approach that produces the lowest response time, high throughput and overall improved performance for the 10 nodes experimented. The improvement in response time, and throughput was sought-after capabilities of cloud computing in load balancing strategy. The result and discussion of the study are presented in this section.

***Presentation of Results of the Developed Load Balancing System***

The load balancing challenge in cloud computing was addressed using a hybrid approach combining PSO, ACO and the Honeybee technique to reconstruct and obtain an improved load balancing system in cloud computing. The result display interface of Microsoft Azure was used to obtain results for the performance evaluation after the load balancing strategy was constructed and was tested with three algorithms: honeybee, PSO, round-robin and two hybridized techniques:

Simulated Annealing Symbiotic Organisms Search (SASOS) and PSO–ACO.

The major resources used in Microsoft Azure cloud:

i) Node: This panel allows users to create the number of nodes needed at a given time
ii) Network Server: This is an attribute that allows users to create and design the type of network server needed for a particular job
iii) Virtual Machine (VM): This platform allows users to create the numbers of VM needed for a project and also allows the user to allocate the VM necessary applications.
iv) Load Balancer: This platform allows the user to develop a load balancer and incorporate it into the system.
v) Network: This attribute allows the user to develop the type of network (TCP/UDP) needed for the job.
vi) Result Storage: This platform stores and displays the results.

The developed load-balancing system (P-ACOHONEYBEE) was constructed to perform self-partitioning and clustering of resources within the cloud computing, in which, twelve virtual machines were partitioned into three clusters (web tier, business tier and data tier) according to their storage area. The system groups all VMs with less than 100 GB storage area as web tier and VMs with storage area of 100 GB and less than 500 GB as business tier, wherein VMs with 500 GB and above were grouped into the data tier.

***Results Obtained on Response Time for the Developed Load Balancing System***

Results obtained from the implementation of the developed load balancing system in terms of response time are presented in Tab. 1. The response time of the developed strategy was considered during the normal period. The peak period was the period when the network latency was high (4000 mls Time to Live (TTL) and above) and the normal period was the period when the network latency was normal (below 4000 mls TTL). The number of nodes used was varied from 10 to 20 nodes to determine the effect of more nodes on the developed load balancing system.

**Table 1:** Response time at 10 nodes during the normal network period

| Number of Task | HoneyBee (ms) | PSO (ms) | SASOS (ms) | Round-robin (ms) | PSO-ACO (ms) | P-ACOHoneyBee (ms) |
|---|---|---|---|---|---|---|
| 100 | 203 | 203 | 201 | 196 | 198 | 184 |
| 200 | 211 | 210 | 212 | 217 | 208 | 203 |
| 300 | 216 | 212 | 209 | 211 | 209 | 197 |
| 400 | 221 | 217 | 220 | 217 | 217 | 204 |
| 500 | 225 | 225 | 222 | 219 | 221 | 207 |
| 600 | 228 | 227 | 224 | 222 | 222 | 208 |
| 700 | 250 | 253 | 251 | 248 | 242 | 227 |
| 800 | 277 | 277 | 276 | 272 | 272 | 267 |
| 900 | 300 | 299 | 300 | 300 | 294 | 279 |
| 1000 | 310 | 308 | 311 | 313 | 302 | 288 |
| 2000 | 350 | 353 | 354 | 352 | 342 | 335 |
| SUM | 2791 | 2784 | 2780 | 2767 | 2727 | 2599 |

### *4.1 Results and Analysis of Response Time at 10 Nodes*

This section analyzes the results obtained using response time as a metric to evaluate the developed system. This research focuses on the development of a novel load balancing strategy for cloud computing environments using PSO, ACO and honeybee algorithms. A load balancing approach named P-ACOHONEYBEE for a cloud environment was developed to address the issue of uneven distribution of network traffic coming into the computer system on a network side, as well as on a server-side and server overloading and application unresponsiveness. In Tab. 1, the results obtained for the response time of the developed load balancing strategy (P-ACOHONEYBEE) are presented. Ten (10) nodes were used for the peak and normal network period and P-ACOHONEYBEE was tested with other strategies (honeybee, PSO, round-robin, SASOS and PSO-ACO) to evaluate the performance and the results of the evaluation were also presented. During the normal network period, a sum of 2599 ms was achieved for P-ACOHONEYBEE and 2791 ms for honeybee, 2784 ms for PSO, 2767 ms for round-robin, 2780 ms for SASOS and 2727 ms for PSO-ACO. This revealed that P-ACOHONEYBEE produced the lowest response time among all the strategies for both the normal period. The results of response time at 10 Nodes during the normal network period is depicted in Fig. 2, wherein all the six expressions were plotted against time (ms). P-ACOHONEYBEE shows outstanding performance among the strategies balancing in the cloud computing environment according to this research.

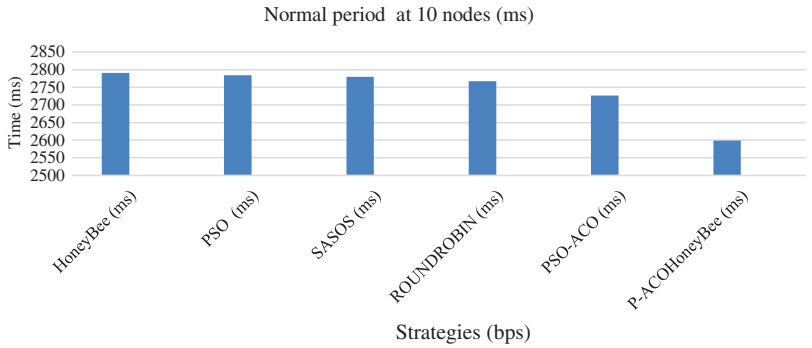


**Figure 2:** Summary of response time results

### *4.2 Results and Analysis of Throughput at 10 Node*

In this section, we analyzed the results obtained using throughput as a metric in the developed system. The research focuses on the development of a novel load balancing strategy for cloud computing environments using PSO, ACO and honeybee algorithms. A load balancing approach named P-ACOHONEYBEE for a cloud computing environment was developed to address the issue of uneven distribution of network traffic coming into the computer system both on a network side as well as on a server-side and server overloading and application unresponsiveness. Results obtained for the throughput of the developed load balancing strategy (P-ACOHONEYBEE) are presented in Tab. 2. A total of 10 nodes were used for the peak and normal network period and P-ACOHONEYBEE was tested with other aforementioned strategies to evaluate the performance, the results of the evaluation are also presented in Tab. 2. The results obtained for throughput of the developed load balancing strategy (P-ACOHONEYBEE) and all other strategies for the normal network period at 10 nodes are presented in Fig. 3. Results show that P-ACOHONEYBEE produced the highest throughput for the normal network period at 10. P-ACOHONEYBEE at 10 nodes produced the highest with a sum

of 7484 and 7485 bps, followed by honeybee which produced a sum of 745 1 and 7438 bps. Time (ms) is plotted against the strategies. P-ACOHONEYBEE shows outstanding performance among the strategies, this makes P-ACOHONEYBEE a novel approach for the load balancer in the cloud environment.

**Table 2:** Throughput results at 10 nodes during the normal network period

| Number of Tasks | HoneyBee (bps) | PSO (bps) | SASOS (bps) | Round-robin (bps) | PSO-ACO (bps) | P-ACOHoneyBee (bps) |
|---|---|---|---|---|---|---|
| 100 | 95 | 91 | 90 | 85 | 91 | 97 |
| 200 | 197 | 195 | 193 | 191 | 193 | 197 |
| 300 | 297 | 297 | 296 | 293 | 288 | 300 |
| 400 | 395 | 394 | 394 | 390 | 390 | 398 |
| 500 | 496 | 493 | 489 | 484 | 489 | 500 |
| 600 | 596 | 591 | 589 | 585 | 593 | 599 |
| 700 | 695 | 692 | 687 | 682 | 690 | 698 |
| 800 | 795 | 792 | 789 | 784 | 787 | 799 |
| 900 | 895 | 892 | 889 | 886 | 889 | 899 |
| 1000 | 995 | 993 | 991 | 987 | 991 | 997 |
| 2000 | 1995 | 1995 | 1991 | 1990 | 1986 | 1998 |
| SUM | 7451 | 7425 | 7398 | 7357 | 7387 | 7482 |

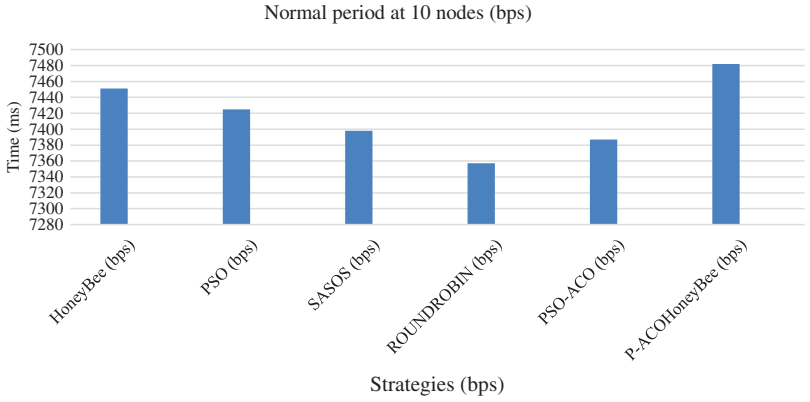


**Figure 3:** Summary of throughput results

We have been able to implement a novel load balancer building on the existing approaches, the novelty of this research lines on the ability to come up with an approach that consumes better computational resources than the existing approaches in the cloud environment. The state-of-the-art evaluation parameters in the cloud computing environment (response time and throughput) were used to measure the performance of the developed approach. The response time for honeybee, PSO, SASOS, Round-robin, PSO-ACO and P-ACOHONEYBEE was 2791, 2784, 2780, 2767, 2727, and 2599 (ms) respectively. Using the response time as a metric for the evaluation, the novel P-ACOHONEYBEE approach developed is better by 128 (ms) than PSO-ACO, which is close to it. In the same vein, when

the throughput is used for the evaluation, honeybee, PSO, SASOS, Round-robin, PSO-ACO and P-ACOHONEYBEE achieved 7451, 7425, 7357, 7387 and 7482 (bps) respectively. Using the throughput as a metric for the evaluation, the P-ACOHONEYBEE approach developed is better by 31 (bps) than honeybee which is close to it. Because an approach with lower response time and high throughput is required for the effectiveness and efficiency of cloud computing operation, the P-ACOHONEYBEE approach is good, as it has performed better than the existing approaches in this research.

## 5 Conclusion and Future Scope

This research helps in realize an efficient and effective load balancing approach highly suitable and achievable for cloud computing which in turn facilitates migration optimization in distributed processing systems and implementation. The research focuses on the development of a novel load balancing strategy for cloud computing environments using PSO, ACO and honeybee algorithms. A load balancing approach named P-ACOHONEYBEE for a cloud computing environment was developed to address the issue of uneven distribution of network traffic coming into the computer system on a network, side as well as on a server-side and server overloading and application unresponsiveness. The developed strategy was created in the Microsoft Azure cloud environment to maximum resource usage, throughput, and reaction time and are all achieved through dynamic resource scheduling that is scalable and reliable. The performance of the developed load balancing system was evaluated alongside the existing algorithms; honeybee, PSO-ACO, SASOS, PSO, and round-robin using response time and throughput as metrics. The results from the implementation showed that a reduction in the response time and throughput was observed when the developed load balancing system is applied in the cloud computing environment. This implies that in terms of the aforementioned metrics, the P-ACOHONEYBEE is computationally efficient, effective and superior to the other methods. The P-ACOHONEYBEE produced the lowest response time and highest throughput than other systems. This is an interpretation that P- ACOHONEYBEE was able to send more successful data per second and it also confirms that it is computationally efficient and superior to the other methods. Conclusively, this research has helped in managing the imbalance drawback by maximizing resource utilization, maximizing throughput, reducing response time, dynamic resource scheduling with scalability and reliability hence showing the novelty of the developed strategy, it has equally guarantee the regulation of traffic and workload. This in turn enhances the performance of cloud computing and other distributed computing adopting the solution in practice.

### Future Scope

In the near future, we plan to extend the testing of P-ACOHONEYBEE protocol by comparing normal time or period with peak period with other nature-inspired techniques such as Firefly, Cockroach Swarm Optimization algorithm, and evolutionary computing to optimize load balancing in cloud computing.

**Acknowledgement:** The authors would like to appreciate the support of Taif University Researchers Supporting Project number (TURSP-2020/211), Taif University, Taif, Saudi Arabia.

**Funding Statement:** Taif University Researchers are supporting project number (TURSP-2020/211), Taif University, Taif, Saudi Arabia.

**Conflicts of Interest:** The authors declare that there is no conflict of interest regarding the publication of the paper.

## References


[1] S. O. Olabiyisi, T. M. Fagbola and R. S. Babatunde, "An exploratory study of cloud andubiquitous computing systems," *World Journal of Engineering and Pure and Applied Sciences*, vol. 2, no. 5, pp. 814–825, 2012.

[2] G. K. Neha and V. B. Bhagat, "An systematic overview on cloud computing and load balancing in the cloud," *International Journal of Engineering Research & Technology (IJERT)*, vol. 2, no. 11, pp. 201–208, 2013.

[3] S. A. Ajagbe, A. O. Adesina and J. B. Oladosu, "Empirical evaluation of efficient asymmetric encryption algorithms for the protection of electronic medical records (EMR) on web application," *International Journal of Scientific and Engineering Research*, vol. 10, no. 5, pp. 848–871, 2019.

[4] J. Wan, B. Chen, S. Wang, M. Xia, D. Li *et al.,* "Fog computing for energy-aware load balancing and scheduling in smart factory," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4548–4556, 2018.

[5] M. Laroui, B. Nour, H. Moungla, M. A. Cherif, H. Afifi *et al.,* "Edge and fog computing for IoT: A survey on current research activities & future directions," *Computer Communications*, vol. 180, no. 1, pp. 210–231, 2021.

[6] H. E. Obinna and B. Seren, "A systematic mapping study on soft computing techniques to the cloud environment," *Procedia Computer Science*, vol. 120, no. 45, pp. 31–38, 2017.

[7] L. Jian, M. Tinghuai, T. Meili, S. Wenhai and J. Yuanfeng, "Improved FIFO scheduling algorithm based on fuzzy clustering in cloud computing," *Information*, vol. 8, no. 5, pp. 1–13, 2017.

[8] X. Zhang, W. Zhang, W. Sun, X. Sun and S. K. Jha, "A robust 3-D medical watermarking based on wavelet transform for data protection," *Computer Systems Science & Engineering*, vol. 41, no. 3, pp. 1043–1056, 2022.

[9] I. Bamimore and S. A. Ajagbe, "Design and implementation of smart home for security using radio frequency modules," *International Journal of Digital Signals and Smart Systems (Inderscience Journal)*, vol. 4, no. 4, pp. 286–303, 2020.

[10] K. Boonhatai and K. Warangkhana, "Enhancing of artificial Bee Colony algorithm for virtual machine scheduling and load balancing problem in cloud computing," *International Journal of Computational Intelligence Systems*, vol. 13, no. 1, pp. 496–510, 2020.

[11] S. G. Domanal and G. R. M. Reddy, "Optimal load balancing in cloud computing by efficient utilisation of virtual machines," *Communication Systems and Networks*, vol. 10, no. 4, pp. 1–4, 2014.

[12] S. A. Ajagbe, A. O. Adesina, T. J. Odule and O. Aiyeniko, "Evaluation of computing resources consumption of selected symmetric-key algorithms," *The Journal of Computer Science and Its Applications*, vol. 26, no. 2, pp. 64–76, 2019.

[13] P. M. Shameem and R. S. Shaji, "A methodological survey on load balancing techniques in cloud computing," *International Journal on Advances in Cloud Computing*, vol. 23, no. 15, pp. 3801–3812, 2013.

[14] G. Sarmila, N. Gnanambigai and P. Dinadayalan, "Survey on fault tolerant and load balancing algorithms in Cloud Computing," *Electronics and Communication Systems*, vol. 15, no. 11, pp. 1715–1720, 2015.

[15] S. Gang, V. Anand, Y. Hong-Fang, L. Dan and L. Lemin, "Optimal provisioning for elastic service oriented virtual network request in Cloud Computing," in *Global Communications Conf. (GLOBECOM)*, Anaheim, CA, USA, pp. 2517–2522, 2012.

[16] N. Chopra and S. Singh, "HEFT based workflow scheduling algorithm for cost optimisation within deadline in hybrid Clouds," *Computing, Communications and Networking Technologies*, vol. 7, no. 3, pp. 1–6, 2013.

[17] S. V. Pius and S. Suresh, "A novel algorithm of load balancing in distributed filesystem for Cloud," *Innovations in Information, Embedded and Communication Systems (ICIIECS)*, Coimbatore, India, vol. 2015, pp. 1–4, 2015.

[18] L. Singh and S. Singh, "A survey of workflow scheduling algorithms and research issues," *International Journal of Computer Applications*, vol. 74, no. 15, pp. 21–28, 2013.

[19] R. Gao and J. Wu, "Dynamic load balancing strategy for cloud computing with ant colony optimization," *Future Internet*, vol. 7, no. 3, pp. 465–483, 2015.
[20] S. R. Shishira, A. Kandasamy and K. Chandrasekaran, "Survey on Meta heuristic optimization techniques in cloud computing," *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, vol. 12, pp. 1434–1440, 2016.
[21] M. Shorfuzzaman and M. Masud, "Leveraging A multi-objective approach to data replication in cloud computing environment to support big data applications," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 10, no. 3, pp. 418–429, 2019.
[22] L. Nkosi, P. Tarwireyi and M. O. Adigun, "Insider threat detection model for the cloud," in *Proc. of the 2013 Information Security for South Africa*, 14–16 August 2013, Johannesburg, South Africa, pp. 1–8, 2013.
[23] R. B. Chakraborty, M. Pandey and S. S. Rautaray, "Managing computation load on a Blockchain–based Multi–Layered Internet–of–Things Network," in *Int. Conf. on Computational Intelligence and Data Science (ICCIDS 2018)*, Gurugram, India, vol. 132, 2018.
[24] P. M. Rekha and M. Dakshayini, "Dynamic cost-load aware service broker load balancing in virtualization environment," in *Int. Conf. on Computational Intelligence and Data Science*, Gurugram, India, vol. 132, 2018.
[25] S. Tuli, N. Basumatary, S. S. Gill, M. Kahani, R. C. Arya *et al.,* "HealthFog: An ensemble deep learning based smart healthcare system for automatic diagnosis of heart diseases in integrated IoT and fog computing environments," *Future Gener. Comput. Syst*, vol. 104, pp. 187–200, 2019.
[26] S. Bitam, S. Zeadally and A. Mellouk, "Fog computing job scheduling optimization based on bees swarm," *Enterprise Information Systems (EIS)*, vol. 12, no. 4, pp. 373–397, 2018.
[27] A. Aggarwal, P. Dimri, A. Agarwal, M. Verma, H. A. Alhumyani *et al.,* "IFFO: An improved fruit fly optimization algorithm for multiple workflow scheduling minimizing cost and makespan in cloud computing environments," *Mathematical Problems in Engineering*, vol. 2021, no. 3, pp. 9, 2021.
[28] H. Topcuoglu, S. Hariri and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 13, no. 3, pp. 260–274, 2020.
[29] T. Choudhari, M. Moh and T.-S. Moh, "Prioritized task scheduling in fog computing," in *Annual ACM Southeast Conf. (ACMSE)*, USA, pp. 22, 2018.
[30] D. Hoang and T. D. Dang, "FBRC: Optimization of task scheduling in Fog-based Region and Cloud," in *IEEE Int. Conf. on Big Data Science and Engineering, and IEEE Int. Conf. on Embedded Software and Systems (Trustcom/BigDataSE/ICESS)*, Sydney, Australia, vol. 2017, pp. 1109–1114, 2017.
[31] J. Li, S. Su, X. Cheng, Q. Huang and Z. Zhang, "Cost-conscious scheduling for large graph processing in the cloud," in *IEEE Int. Conf. on High Performance Computing and Communications (HPCC)*, Banff, Alberta, Canada, vol. 2021, pp. 808–813, 2011.
[32] S. Kabirzadeh, D. Rahbari and M. Nickray, "A hyper heuristic algorithm for scheduling of fog networks," in *IEEE Conf. of Open Innovations Association (FRUCT)*, Helsinki, vol. 2017, Finland pp. 148–155, 2017.
[33] M. Nouiri, A. Bekrar, A. Jemai, S. Niar and A. C. Ammari, "An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem," *Journal of Intelligent Manufacturing*, vol. 29, no. 3, pp. 603–615, 2018.
[34] L. Huang, G. Li, J. Wu, L. Li, J. Li *et al.,* "Software-defined QoS provisioning for fog computing advanced wireless sensor networks," *IEEE Sensors*, Orlando, FL, USA, vol. 2016, pp. 1–3, 2016.
[35] N. Kiran, X. Liu, S. Wang and C. Yin, "Optimising resource allocation for virtual network functions in SDN/NFV-enabled MEC networks," *IET Communications*, vol. 15, no. 13, pp. 1710–1722, 2021.
[36] D. Qi, S. Shen and G. Wang, "Towards an efficient VNF placement in network function virtualization," *Comput Commun. (ComCom)*, vol. 138, no. 3, pp. 81–89, 2019.
[37] I. M. Ali, K. M. Sallam, N. Moustafa, R. Chakraborty, M. J. Ryan *et al.,* "An automated task scheduling model using non-dominated sorting genetic algorithm II for fog-cloud systems," *IEEE Transactions on Cloud Computing*, vol. 2020, pp. 1, 2020.

[38] C. S. Pawar and R. B. Wagh, "Priority based dynamic resource allocation in cloud computing with modified waiting queue," in *IEEE Int. Conf. on Intelligent Systems and Signal Processing (ISSP)*, Vallabh Vidyanagar, Anand, Gujarat India, vol. 2013, pp. 311–316, 2013, http://dx.doi.org/10.1109/ISSP.2013.6526925.
[39] C. S. Nandyala and H.-K. Kim, "From cloud to fog and IoT-based real-time Uhealthcare monitoring for smart homes and hospitals," *International Journal of Smart Home*, vol. 10, no. 2, pp. 187–196, 2016.
[40] A. V. Lakra and D. K. Yadav, "Multi-objective tasks scheduling algorithm for cloud computing throughput optimization," *Procedia Comput. Sci.*, vol. 48, pp. 107–113, 2015.
[41] V. Cardellini, V. Grassi, F. L. Presti and M. Nardelli, "On QoS-aware scheduling of data stream applications over fog computing infrastructures," in *IEEE Symp. on Computers and Communication (ISCC)*, Golden Bay, Larnaca, Cyprus, pp. 271–276, 2015.
[42] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin *et al.,* "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 3860–3873, 2016.
[43] W. Hou, Z. Ning and L. Guo, "Green survivable collaborative edge computing in smart cities," *IEEE Transactions on Industrial Informatics (TII)*, vol. 14, no. 4, pp. 1594–1605, 2018.
[44] H. Mahdizadeh, "Designing a smart method for load balancing in cloud computing," *International Journal of Mechatronics, Electrical and Computer Technology (IJMEC)*, vol. 7, no. 24, pp. 3314–3324, 2017.
[45] J. He, J. Wei, K. Chen, Z. Tang, Y. Zhou *et al.,* "Multitier fog computing with large-scale IoT data analytics for smart cities," *IEEE Internet of Things Journal (IoT-J)*, vol. 5, no. 2, pp. 677–686, 2018.
[46] M. Mtshali, H. Kobo, S. Dlamini, M. Adigun and P. Mudali, "Multi-objective optimization approach for task scheduling in fog computing," in *2019 Int. Conf. on Advances in Big Data, Computing and Data Communication Systems (icABCD)*, South Africa, pp. 1–6, 2019.
[47] P. Somwang, "Efficient load balancing for cloud computing by using content analysis," *International Journal of Communication Networks and Information Security (IJCNIS)*, vol. 12, no. 2, pp. 242–247, 2020.
[48] L. Sliwko, "Intelligent Load Balancing in Cloud Computer Systems, a PhD thesis awarded by the University of Westminster," 2019.
[49] Y. Mao, D. Ren and X. Chen, "Adaptive load balancing algorithm based on prediction model in cloud computing," in *ICCC'13*. China: ACM Wuhan, 2013.
[50] K. M. Abiodun, J. B. Awotunde, R. O. Ogundokun, M. Sanjay and E. Adeniyi, "Applicability of MMRR load balancing algorithm in cloud computing," *International Journal of Computer Mathematics: Computer Systems Theory*, vol. 6, no. 1, pp. 7–20, 2020.