Hindawi
Applied Computational Intelligence and Soft Computing
Volume 2022, Article ID 3183701, 13 pages
https://doi.org/10.1155/2022/3183701

Research Article

Differential Grey Wolf Load-Balanced Stochastic Bellman Deep Reinforced Resource Allocation in Fog Environment

S. V. Nethaji [1,2] **and M. Chidambaram** [3]

[1]*PG & Research Department of Computer Science, Rajah Serfoji Government College (Autonomous), (Affiliated to Bharathidasan University, Tiruchirappalli), Thanjavur, India*
[2]*PG Department of Computer Science, M. R. Government Arts College, (Affiliated to Bharathidasan University, Tiruchirappalli), Mannargudi, India*
[3]*PG & Research Department of Computer Science, Rajah Serfoji Government College (Autonomous), (Affiliated to Bharathidasan University, Tiruchirappalli), Thanjavur, India*

Correspondence should be addressed to S. V. Nethaji; nethajisv@gmail.com

Received 31 May 2022; Accepted 1 August 2022; Published 19 August 2022

Academic Editor: Agostino Forestiero

Fog computing is becoming a dynamic and sought-after computing prototype for Internet of Things (IoT) application deployments. It works in conjunction with the cloud computing environment. Load balancing, which is employed by IoT applications when deciding, which fog or cloud computing nodes to use, is one of the most critical components for enhancing resource efficiency and avoiding problems like overloading or underloading. However, for IoT applications, ensuring that all CPU nodes are evenly distributed in terms of latency and energy utilization remains a challenge. To solve these issues, this work introduces Differential Grey Wolf (DGW) load balancing with stochastic Bellman deep reinforced resource optimization (DGW-SBDR) in fog situations. A Differential Evolution-based Grey Wolf Optimization algorithm based on load balancing has been developed for optimal resource management. The Grey Wolf Optimization algorithm, which employs differential evolution, assigns jobs to virtual machines based on user demands (VMs). In the event of an overutilized VM pool, a grey wolf optimization strategy based on differential evolution can detect both under and overutilized VMs, allowing for smooth transit between fogs. This step disables a number of virtual machines in order to reduce latency. In a Stochastic Gradient and Deep Reinforcement Learning-based Resource Allocation Model, a stochastic gradient bellman optimality function and Deep Reinforcement Learning are integrated for optimal resource allocation. According to the proposed method, QoS may be supplied to end-users by reducing energy consumption and better managing cache resources utilizing stochastic gradient bellman optimality.

1. Introduction

Fog computing, also known as edge computing, is a type of distributed computing that connects numerous IoT sensors or devices to a cloud. This is due to the fact that many of these IoT sensors or devices generate a large amount of raw data. Rather than sending all of this data to cloud-based servers for further processing, the concept behind it is to do as much processing with it as possible to reduce bandwidth costs and hence reduce latency between input and response.

In [1] a strategy for load balancing in fog computing environments called Dynamic Energy Efficient Resource Allocation (DEERA) [2] was suggested. To begin, user tasks were sent to the task manager to be completed. After that, the resource information provider used Cloud Data Centers to register the resources. After that, the information for both tasks and resources was sent to the resource scheduler. The resource scheduler makes further arrangements for the available resources based on their use.

Finally, tasks and resources from the resource scheduler were assigned to resources according to an ordered list of

tasks. The results of a successful job execution were sent to the associated user via a resource engine, reducing both energy usage and computing costs. The latency associated with load balancing in fog computing environments was not highlighted, despite improvements in both energy consumption and compute cost. To address this issue, we present the Differential Evolution-based Grey Wolf Optimization algorithm [3, 4], which ensures optimal load balancing with minimal latency through node update, relocation, and migration [5].

The Internet of Things (IoT) allows for the coordination and balancing of a large number of Internet-connected devices. However, due to processing congestion, an overload on a network link connected to an IoT gateway may result in inaccessibility in IoT applications. To solve these issues, [6] provided a solution for the requirements to implement load balancing for the Fog of Things via programmability via the use of Software Defined Networks (SDN).

The provision of processing, storage, and IoT services at edge devices was ensured with a minimum response time based on the FoT environment, as well as a better reduction in the average time between requests. Despite the fact that response time and storage were reduced, the energy utilized throughout the operation was not targeted. The Stochastic Bellman Gradient Deep Reinforcement Learning-based Resource Allocation algorithm was created to address this issue. The approach, which combines the Stochastic Bellman Gradient optimality function with Deep Reinforcement Learning, not only improves the makespan but also minimizes the amount of energy used by the stochastic gradient function.

The major goal of this study is to present a new load-balanced resource allocation model in the fog that efficiently accomplishes load balancing and resource allocation separately. Because of its multi-factor, i.e., optimization and deep learning model, this optimization model solves the shortcomings of previous load balancing methods in fog environments. This paper's contributions also include the following:

To propose a unique Grey Wolf Optimization algorithm [4, 5] based on differential evolution for minimizing a composite objective function. Using a differential evolution function, our approach can reduce the latency rate during load balancing.

To create a new Stochastic Bellman Gradient Deep Reinforcement Learning-based Resource Allocation method based on the Stochastic Bellman Gradient function and a deep learning model that assures improved optimal load balancing and resource allocation in the fog. In a fog environment, the suggested DGW-SBDR technique enhanced load balancing efficiency, makespan, latency, and energy consumption as performance assessment measures.

The remainder of the paper is structured as follows: Section 2 discusses the various modern scheduling approaches that are available. In Section 3, we spoke about the suggested Differential Grey Wolf load balancing and Stochastic Bellman Deep Reinforced resource optimization (DGW-SBDR) in a fog environment. Section 4 describes the experimental setup and comparative analysis, followed by a detailed explanation of the conclusions in Section 5.

2. Works That Are Related

Over the last several years, the number of IoT devices with sensors has increased significantly. The data from these sensors is obtained by these sensors. Following that, with the help of the raw data gathered, aggregated information is produced with the goal of making an automatic conclusion. Overloaded fog servers, on the other hand, may be unable to maintain Quality of Service (QoS), resulting in increased delay.

In [7] an integrated fog-cloud environment was proposed with the goal of lowering resource allocation costs and minimizing latency. A cooperative three-layer fog-cloud computing environment was built for this purpose. A unique optimization model with a composite goal function was created to decrease bandwidth costs while also ensuring effective load balancing. A Dynamic and Resource Aware Load-Balanced Scheduling Approach for Cloud Computing [DRALBA] was developed in [8] with the goal of reducing average response time while increasing throughput.

Despite significant infrastructure improvements, cloud computing still faces a number of challenges when it comes to load balancing. However, the majority of current techniques analyze only a few QoS measures and overlook other critical aspects. In [9] a load balancing technique called data files type formatting (DFTF) was proposed, which combines an enhanced version of cat swarm optimization (CSO) with a support vector machine (SVM). This resulted in increased throughput as well as a reduction in migration time.

The Internet of Things (IoT) prototype has been viewed as a critical management tool for implementing the smart concept in a variety of areas for reasons of monitoring, control, and management. However, overloaded networks have resulted from the exponential increase in data traffic caused by the rise in IoT-enabled devices.

In [10], a unique Effective Prediction and Resource Allocation Methodology (EPRAM) for the Fog environment was suggested, which is applicable for healthcare applications. Makespan was discovered to be reduced to a greater extent with the help of reinforcement learning. To ensure service quality, another method called Queuing theory-based cuckoo search was developed in [11]. The fog computing environment's challenges and outstanding issues based on reinforcement learning were discussed in [12].

Smart mobile gadgets have become increasingly important in our daily lives over the last few years. Despite the advancements, numerous applications are still experiencing energy consumption issues. Fog computing has recently been presented as an alternative to traditional cloud computing to address the conflict between resource-restricted applications and devices. Fog computing ensures computing potentialities with flexible computation and communication services.

In [13], two alternative resource allocation strategies for hybrid systems were developed. During the decision-making process, the technique considered network load in addition to resource allocation. The overall goal remained the allocation of resources as well as the reduction of time spent in the whole process. This was accomplished by allocating

weights to each optimization criterion; previously, resource allocation policies used predetermined weights; however, the weights were identified in a dynamic way, resulting in increased accuracy.

Fog computing was used to establish a collaborative resource management policy for device communication in [14]. In this study, each mobile end chooses whether to offload through an edge server or through a third-party fog node. With this architecture, an optimal solution algorithm was devised utilizing mixed integer nonlinear programming and based on branch-and-price, reducing computational complexity to a greater extent.

While fog IoT applications are important, there is still a gap in properly utilizing computing resources in a fog environment with increased Quality of Service (QoS) and Quality of Experience (QoE) (QoE). [15] used the greedy method, which not only ensured scalability but also achieved optimality for multitasking applications in a fog environment. [16] used a modified Artificial Echo System-based Optimization approach to solve the work scheduling problem. Maximum makespan time and throughput were supposed to be guaranteed with this configuration.

[17] suggested a dynamic optimization mechanism for IoT fog environments, where offloading decisions are determined in real-time. To be more explicit, a hybrid compute offloading and radio resource allocation approach based on Lyapunov optimization was presented with the goal of minimizing latency and energy usage. However, because of the fog node's compute constraints and storage inefficiency, offloading jobs must be done as efficiently as possible. To achieve this goal, the Globally Optimal Multi-Objective Optimization algorithm for Task Offloading (GOMOTO) was used in conjunction with a network calculus theory [18] for task offloading in a fog environment.

Another unique method, Energy-aware Fog Resource Optimization (EFRO), was developed in [19] with the goal of optimizing devices in a fog environment. To achieve this goal, a heuristic algorithm was created, which lowered not only the cost of energy but also the time it took to complete the task. [20] suggested a fog scheduling strategy that took into account multiple access download times, lowering the average response time.

[21] proposed an integrated iterative optimization on subcarrier that took into account power and trajectory to arrive at the best option. Furthermore, a fairness optimization model was presented to enhance the minimum transmit rate of IoT nodes, resulting in improved transit performance. In [22], a polynomial mutation process based on the Cauchy distribution was developed to solve multiple objective problems, namely, maintaining a tradeoff between makespan and energy consumption. [23] used the Opposition-based Chaotic Whale Optimization Algorithm to examine two QoS criteria, optimizing time and energy usage. [24] used a direct load control technique to ensure load optimization and flexibility.

In response to the aforementioned challenges, a strategy termed Differential Grey Wolf load balancing and Stochastic Bellman Deep Reinforced resource optimization (DGW-SBDR) in a fog environment is proposed in this paper. The next sections provide a more detailed discussion of the suggested method.

3. Methodology

The proposed approach is divided into two parts. The first step is concerned with balancing the load between fog and cloud networks. The second step, on the other hand, efficiently allocates resources between virtual machines in the cloud. The significance of the findings when a user request service or user request task is sent to a cloud network is determined by the IoT application in use. The traditional cloud computing environment is becoming overcrowded, resulting in concerns such as jitter and slowness. As a result, the fog layer was created as a transitional layer between the user and the cloud computing environment. As a result, the first phase of our work will focus on balancing the load between the fog layer and the cloud computing environment using a Grey Wolf Optimization model based on differential evolution.

The second step, on the other hand, skips over the optimization of cloud resource allocation for user-requested tasks. The goal here is to apply a stochastic gradient bellman function to reduce the amount of energy consumed during the process of assigning resources across user-requested tasks, resulting in optimal VM allocation based on deep reinforcement learning. In a fog setting, a block diagram of Differential Grey Wolf load balancing with Stochastic Bellman Deep Reinforced resource optimization (DGW-SBDR) is proposed (see Figure 1).

As shown in Figure 1, the proposed DGW-SBDR method is split into two sections. They are load balancing using Differential Evolution-based Grey Wolf Optimization model and resource allocation using Stochastic Gradient and Deep Reinforcement Learning-based Resource Allocation model. The detailed description of the proposed DGW-SBDR method is detailed in the following sections.

3.1. Differential Evolution-Based Grey Wolf Optimized Load Balancing Model. To balance load between fog and cloud networks, a novel Differential Evolution-based Grey Wolf Optimization model is designed. The Differential Evolution-based Grey Wolf Optimization model detects overutilized fogs, underutilized fogs, and finally performs migration in a significant manner, therefore, balancing load between fog and cloud networks. With this, the latency involved in completing one user request task between source and destination is said to be reduced. Figure 2 shows the block diagram of Differential Evolution-based Grey Wolf Optimization model.

The wolf (i.e., the computing nodes in the fog) has a strong potential to capture prey (i.e., ensuring load distribution in the cloud environment). In order to mimic wolves' internal leadership hierarchy, the wolves are split into four distinct types, "alpha,""beta," "delta," and "omega," where the best individual (i.e., the best computing nodes), second best individual (i.e., the second best computing nodes), and third best individual (i.e., the third best computing nodes)

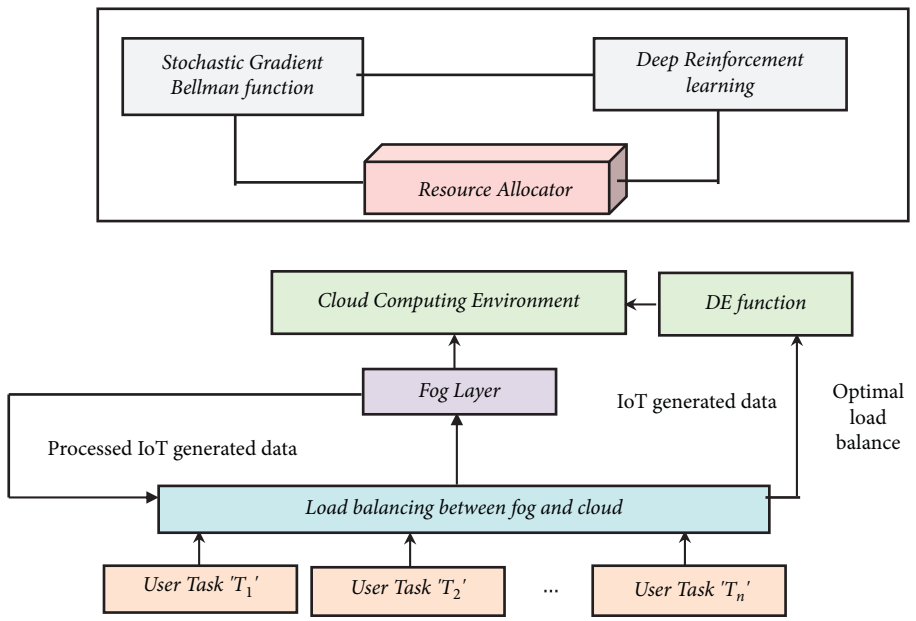

Figure 1: Block diagram of Differential Grey Wolf load balancing and Stochastic Bellman Deep Reinforced resource optimization (DGW-SBDR) method.

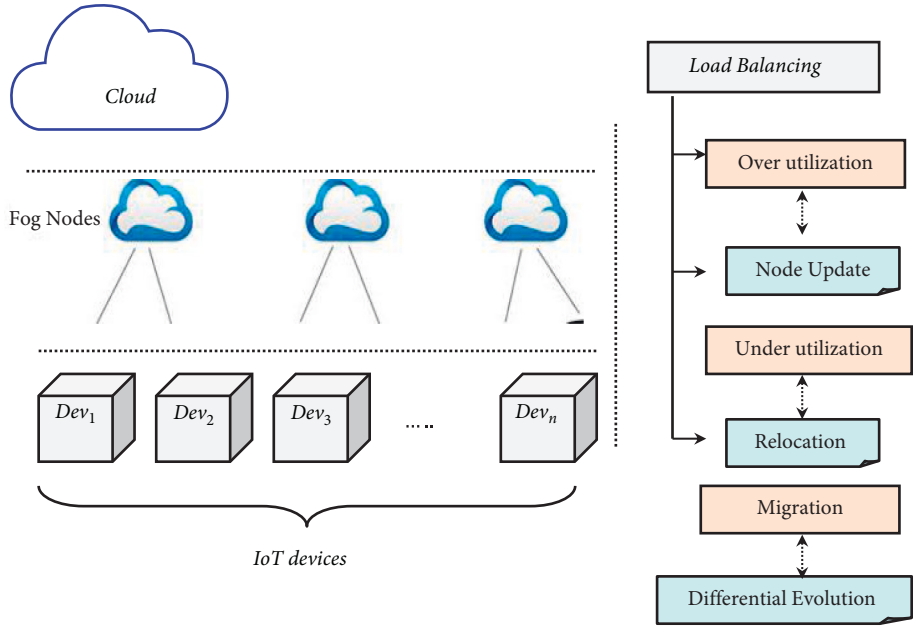

Figure 2: Block diagram of Differential Evolution-based Grey Wolf Optimization.

are considered as "alpha," "beta," and "delta" and the rest of the individuals (i.e., the rest of the computing nodes) are considered as "omega." In the Differential Evolution-based Grey Wolf Optimization, the optimal load balancing is controlled by "alpha," "beta," and "delta." They assist other wolves or the computing nodes in the fog incline to the optimal load balancing to the best area in searching space (i.e., the CC environment). The cloud data centers with "n" numbers of servers "S" are placed as given below.

$$DC = \{DC_1, DC_2, \ldots, DC_n\}, \tag{1}$$

$$S = \{S_1, S_2, \ldots, S_n\}. \tag{2}$$

Then, from the abovementioned placement Equations(1) and (2), the fog network comprises distinct numbers of virtual machines "VMs," with "VMs" processing different numbers of tasks "T" at a time instance "t," mathematically stated as given below.

$$VM = \{VM_1, VM_2, \ldots, VM_n\}, \tag{3}$$

$$T = \{T_1, T_2, \ldots, T_n\}. \tag{4}$$

An overutilized fogs that are receiving too many user requests at a time instance cannot serve all of them at a specified time. As a result, this will accelerate or amplify the response time of fog. To handle this issue and to identify the overutilized fogs, we have employed three wolves, i.e., "alpha," " beta," " delta." Despite the fact that it does require transposing the characteristics of continuous procedures to the discontinuous procedures, by means of the differential evolution, the scale factor is updated, therefore minimizing the probability of falling into the local optimum policy. At this juncture, overutilized fogs are determined. In the optimization process, the positions of wolves or the computing nodes in the fog are updated based on (5) and (6) as given below.

$$\overrightarrow{R} = \overrightarrow{Q} \cdot \overrightarrow{A_p}(t) - \overrightarrow{A}(t), \tag{5}$$

$$\overrightarrow{A}(t+1) = \overrightarrow{A_p}(t) - \overrightarrow{P} \cdot \overrightarrow{R}. \tag{6}$$

From the abovementioned Equations (5) and (6), "t" denotes the "$t - th$" iteration, "$\overrightarrow{P}$" and "$\overrightarrow{Q}$" representing the coefficient vector of computing nodes in the fog, "$\overrightarrow{A_p}$" denotes the position vector of computing prey nodes in the fog and "$\overrightarrow{A}$" denoting the computing node position. The vector "$\overrightarrow{P}$" and "$\overrightarrow{Q}$" are then mathematically formulated as given below.

$$\overrightarrow{P} = 3a \cdot \overrightarrow{\mathrm{Rnd}_1} - \overrightarrow{a}, \tag{7}$$

$$\overrightarrow{Q} = 3 \cdot \overrightarrow{\mathrm{Rnd}_2}. \tag{8}$$

From the abovementioned Equations (7) and (8), the coefficient vector of computing nodes in the fog "a" decreases from "3" to "0" with the increasing iteration size of user request tasks placed in the fog environment, "$\overrightarrow{\mathrm{Rnd}_1}$" and "$\overrightarrow{\mathrm{Rnd}_2}$" denoting the random coefficient vector of computing nodes in the fog. In the proposed Differential Evolution-based Grey Wolf Optimization model, it is assumed that the position of alpha, beta, and delta is probably to be the prey or the optimum load-balanced computing node in fog.

On the other hand, when a fog does not receive several user request tasks and hence the workload is low then it is referred to as the underutilized fog. As far as energy consumption is concerned, the same amount of energy is said to be consumed in case of both the underutilized and overutilized fogs. To address this issue, the virtual machines of these fogs are transferred to the other fogs and are performed for all fogs and owing to this, both the active fog frequency and the energy consumption are also said to be reduced. Moreover, during the iteration searching process, the first best individual, the second best individual, and the third best individual are noted as "alpha," " beta," " delta." However, other computing nodes represented as "omega" relocate their locations, therefore detecting underutilized fogs by mathematically formulating as given below.

$$\overrightarrow{Dis_\alpha} = \left|\overrightarrow{\mathrm{Rnd}_1} \cdot \overrightarrow{A_\alpha} - \overrightarrow{A}\right|, \tag{9}$$

$$\overrightarrow{Dis_\beta} = \left|\overrightarrow{\mathrm{Rnd}_2} \cdot \overrightarrow{A_\beta} - \overrightarrow{A}\right|, \tag{10}$$

$$\overrightarrow{Dis_\gamma} = \left|\overrightarrow{\mathrm{Rnd}_3} \cdot \overrightarrow{A_\gamma} - \overrightarrow{A}\right|. \tag{11}$$

From the abovementioned Equations (9–11), "$\overrightarrow{A_\alpha}$," "$\overrightarrow{A_\beta}$," and "$\overrightarrow{A_\gamma}$" denotes the position vector of computing nodes in the fog for "alpha," " beta," " delta," respectively, with "$\overrightarrow{\mathrm{Rnd}_1}$," "$\overrightarrow{\mathrm{Rnd}_2}$," and "$\overrightarrow{\mathrm{Rnd}_3}$" forming random coefficient vector of computing nodes and "$\overrightarrow{A}$" denoting the position vector of computing node, respectively. Equations (9–11) measure the distance between the position of current computing nodes in the fog and that of individual nodes (i.e., "alpha," " beta," and "delta"). So the final position vectors of the current individual are mathematically obtained as given below.

$$\overrightarrow{A_1} = \overrightarrow{A_\alpha} - \overrightarrow{P_1} - \left(\overrightarrow{\mathrm{Dis}_\alpha}\right), \tag{12}$$

$$\overrightarrow{A_2} = \overrightarrow{A_\beta} - \overrightarrow{P_2} - \left(\overrightarrow{\mathrm{Dis}_\beta}\right), \tag{13}$$

$$\overrightarrow{A_3} = \overrightarrow{A_\gamma} - \overrightarrow{P_3} - \left(\overrightarrow{\mathrm{Dis}_\gamma}\right). \tag{14}$$

Finally, upon overutilization of the fog, migration between VMs is done. The migration process is performed in such a manner that a balance between the fogs is said to be ensured. This is done in our work by employing Differential Evolution (DE) function. The most prominent feature of differential evolution is mutation operation. During the migration between VM, two differences in weight are added to the individual computing node to achieve its dissimilarity. The basic difference element of DE is the dissimilarity vector of the parents, and each vector comprises two distinct individual fog nodes "$(VM^t_{Rnd_1}, VM^t_{Rnd_2})$" and the dissimilar vector to achieve migration between fog nodes is defined as follows:

$$\mathrm{Diff}_{Rnd12} = VM^t_{Rnd_1} - VM^t_{Rnd_2}. \tag{15}$$

With the above handling of overutilization, underutilization, and migration between virtual machines optimal load balancing is said to be ensured. The pseudo-code representation of Differential Evolution-based Grey Wolf Optimization is given below.

As given in the abovementioned Differential Evolution-based Grey Wolf Optimization algorithm with the objective of reducing the latency involved during the load balancing process, in this work is based on the principle of imitating the behavior of grey wolves (i.e., the computing nodes in fog) to hunt (i.e., ensuring load balance) in a cooperative (i.e., optimal manner). According to this principle, overutilization based on node update, underutilization by means of relocation, and migration using Differential Evolution

Input: Dataset "DS" task "$T = \{T_1, T_2, \ldots, T_n\}$," fog nodes "$F = \{F_1, F_2, \ldots, F_n\}$"
Output: Latency-minimized optimal load balancing
(1) Initialize "$\overrightarrow{\text{Rnd}_1}$," "$\overrightarrow{\text{Rnd}_2}$," "a"
(2) Begin
(3) For each dataset "DS" with task "T" and fog nodes "F"
(4) Formulate cloud data centers with "n" numbers of servers "S" as in equations (1) and (2)
(5) Model "VMs" processing different numbers of tasks "T" as in equations (3) and (4)
(6) Update positions of wolves or the computing nodes by handling overutilization of fog as in equations (5) and (6)
(7) Handle under-utilized fog detection using equations (9–11)
(8) Estimate the final position vectors of the current individual as in equations (12–14)
(9) Handle migration between virtual machines as in equation (15)
(10) Return optimal load balancing fog computing nodes
(11) End for
(12) End

Algorithm 1: Differential Evolution-based Grey Wolf Optimization.

function is adopted separately. With these three different operations, latency-minimized optimal load balancing is ensured.

3.2. Stochastic Gradient and Deep Reinforcement Learning-Based Resource Allocation Model. With the discussion of the abovementioned latency-minimized and optimal load balancing mechanism, the next step remains in ensuring smooth and robust allocation of resources between computing nodes in a fog environment. This section introduces a significant Stochastic Gradient and Deep Reinforcement Learning-based resource allocation and task scheduling in a fog environment. The proposed optimal resource allocation problem is incorporated into a stochastic gradient model with the objective of completing as many fog node users' requested tasks as possible while reducing task completion latency and energy consumption via state space, action space, and reward function. Figure 3 shows the block diagram of the Stochastic Gradient and Deep Reinforcement Learning-based Resource Allocation model.

As shown in Figure 3, let us consider that an agent (i.e., the resource allocator) interacts with the fog environment. Then at each time instance, the agent or the load balancer observes the fog environment and acquires the following input from the fog environment "$Inp = \{DS^i, WT^i, QL^i\}$."The elements of the input are then mathematically described in detail as given below.

$$DS^i = \left\{DS^i_{T,1}, DS^i_{T,2}, \ldots, DS^i_{T,n}\right\}. \tag{16}$$

From the abovementioned Equation (16), "$DS^i_{T,n}$" denotes the vector of length "n" consisting of the data size "DS" of the user requested tasks "$T, 1$" to be processed via the load balancer or the agent.

$$WT^i = \left\{WT^i_{T,1}, WT^i_{T,2}, \ldots, WT^i_{T,n}\right\}. \tag{17}$$

From (17), "$WT^i_{T,n}$" denotes the vector of length "n" consisting of the waiting time "WT" of the user requested tasks "$T, 1$" to be processed via the load balancer or the agent.

$$QL^i = \left\{QL^i_{T,1}, QL^i_{T,2}, \ldots, QL^i_{T,n}\right\}. \tag{18}$$

Finally, from the abovementioned Equation (18), "$QL^i_{T,n}$" denotes the vector of length "n" denoting the queue length "QL" of the buffer. After observing the current state "S" at time instance "t", appropriate action has to be taken whether to offload the user-requested task or to execute it locally. For the case scenario of offloading, the effective execution either on host fog or neighbor fog has to be identified.

Let us consider an action "$A(t) = \left\{Y^i_{loc}, Y^F_{ij}, Y^F_{ijk}, \in (0, 1)\right\}$."If the user requested task generated by the end device of fog node "F_i" is executed locally then "$Y^i_{loc} = 1$" otherwise "$Y^i_{loc} = 0$."On the other hand, if user requested a task generated by the end device of fog node "F_i," is executed on host node then "$Y^F_{ij} = 1$" otherwise "$Y^F_{ij} = 0$". On contrary, if user requested a task generated by end device of fog node "F_i", is executed by the neighbor "F_k", then "$Y^F_{ijk} = 1$" otherwise "$Y^F_{ijk} = 0$".

Finally, the design of the reward function is made by the load balancer or the agent by learning an optimal policy with the objective of maximizing the earned reward. The reward function is designed in such a manner that can reduce the overall energy consumption. Then, at a time instance "t," the progressive reward is measured by totaling all the obtained rewards by each agent or load balancer and is formulated as given below.

$$R(t) = \sum_{i=1}^{n} R_i(t) + \alpha E_i(t). \tag{19}$$

From the abovementioned Equation (19), "$R(t)$" represents the received reward at "$i - th$" iteration on time instance "t" and "α" denotes the assigned significance to energy consumption "E_i". The significance value to energy consumption is derived based on the Stochastic Bellman Gradient Optimality function as given below.

$$OA(S, A) = \left[R\left(S_t, A_t\right) + \beta \text{Max} OA\left(S'_t, A'_t\right)\right]. \tag{20}$$

With the derived optimality function given above, concentrating on the minimization of the energy

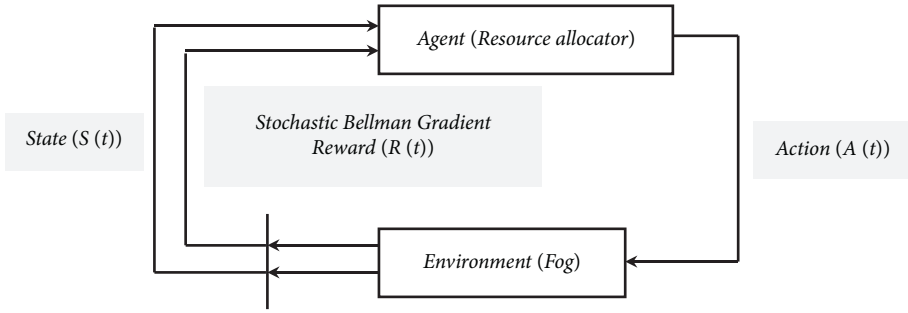

Figure 3: Block diagram of Stochastic Gradient and Deep Reinforcement Learning-based Resource Allocation model.

Input: Dataset "DS," task "$T = \{T_1, T_2, \ldots, T_n\}$," fog nodes "$F = \{F_1, F_2, \ldots, F_n\}$"
Output: Energy minimized optimal resource allocation
(1) Begin
(2) For each dataset "DS" with task "T" and fog nodes "F"
//State space
(3) Load balancer acquires the input from fog environment "Inp = $\{DS^i, WT^i, QL^i\}$"
(4) Mathematically formulate data size as in equation (16)
(5) Mathematically formulate waiting time as in equation (17)
(6) Mathematically formulate queue length as in equation (18)
//Action space
(7) For each action "A" with the consolidated state "S"
(8) If task "T" generated by fog node "F_i" is executed locally
(9) Then "$Y^i_{loc} = 1$"
(10) Else "$Y^i_{loc} = 0$"
(11) End if
(12) If Task "T" generated by fog node "F_i" is executed on the host node
(13) Then "$Y^F_{ij} = 1$"
(14) Else "$Y^F_{ij} = 0$"
(15) End if
(16) If Task "T" generated by fog node "F_i" is executed by neighbor
(17) Then"$Y^F_{ijk} = 1$"
(18) Else "$Y^F_{ijk} = 0$"
(19) End if
//Reward function
(20) For each action "A" with the consolidated state "S" and task "T" generated by fog node "F_i"
(21) Total all the obtained rewards as in equation (19)
(22) Measure stochastic bellman gradient optimality function as in equation (20)
(23) End for
(24) End for
(25) End

Algorithm 2: Stochastic Bellman Gradient Deep Reinforcement Learning-based Resource Allocation.

consumption, resource allocation to the corresponding user-requested tasks are made in a significant manner. The pseudo-code representation of Stochastic Bellman Gradient Deep Reinforcement Learning-based Resource Allocation is given below.

As given in the above algorithm with the objective of minimizing the energy consumption during resource allocation between fog nodes by the load balancer, a novel Deep Reinforcement Learning model is designed. First, the state space representing the user requested task, time is obtained. Second, the action space concerning the Task "T" generated by fog node is acquired. Finally, the Stochastic Bellman Gradient function is applied to the reward function for ensuring optimal resources in an energy-efficient manner.

4. Experimental Evaluation and Results

This section presents the simulation environment settings of the proposed method and evaluates the efficiency of the proposed optimal load-balanced-based resource allocation method in comparison with other benchmark methods, Dynamic Energy Efficient Resource Allocation (DEERA) [1] and load balancing for FoT-Gateways [6] over different performance metrics using Personal Cloud Dataset obtained

from https://cloudspaces.eu/results/datasets [25]. Specifically, this section is divided into two subsections. The first subsection contains information pertaining to the dataset description. In addition, the second subsection discusses the proposed methods analysis and key findings. For the simulation environment, Java Language and iFogSimsimulator that ensures the modeling and to measure load balancing and resource management across fog and cloud resources under different scenarios are presented.

4.1. Simulation Environment and Dataset Description. The iFogSimsimulator includes user interface structures, IoT services, resources, and network applications. In an iFogSimsimulator, a number of user-requested tasks from the edge layer are given to the optimal fog nodes for performing task scheduling in a distributed fog environment using NEC personal cloud trace. The NEC dataset integrates two sources of information, i.e., from the storage layer and sharing interactions. The storage layer identifies and describe files (size, extension), as well as the file owner and the container/folder where it is stored. Next, the sharing interactions contain log lines sharing interactions across users and information about shared files obtained from March 7th 2013 to September 9th 2015. Table 1 Storage Layer Description given below lists the storage layer description and Table 2 given below lists the column field description.

4.2. Analysis and Findings

4.2.1. Performance Analysis of Load Balancing Efficiency and Findings. The first and foremost parameter to be measured for analyzing load balancing in fog is the load balancing efficiency. To be more specific, load balancing efficiency refers to the percentage ratio of user request tasks distributed to fog nodes to the total number of user request tasks. This is mathematically formulated as given below.

$$Eff_{LB} = \left[\frac{T_{DE}}{n}\right] * 100. \tag{21}$$

From Equation (21), load balancing efficiency "Eff_{LB}" is measured based on the requests distributed efficiently "T_{DE}" to the actual user requests "n" in the queue. The load balancing efficiency is measured in terms of percentage (%). The results for performance metrics of load balancing efficiency are provided in Table 3. The simulation results demonstrate that DGW-SBDRmethod improves the load balancing efficiency as compared to DEER [1] and load balancing for FoT-Gateways [6].

Figure 4 displays the load balancing efficiency of three separate user-requested activities supported by fog resources. The x-axis shows the amount of user-requested biosensor tasks to be processed, while the y-axis shows their load balancing in percentage (percent). It displays how different tasks utilize fog resources. DGW-SBDR shows greater load balancing efficiency in the highlighted environment than [1, 6]. The improvement was attributed to using Differential Evolution-based Grey Wolf Optimization to balance host use amongst fog nodes. By employing the Differential Evolution function, this approach determines the genuine global solution (optimal host utilization assuring load balancing) regardless of cloud user request tasks, with quick convergence between under and overutilization. DGW-SBDR enhanced load balancing efficiency by 2% compared to [1] and 6% compared to [6].

4.2.2. Performance Analysis of Makespan and Findings. The second factor of paramount during load balancing with optimal resource allocation is the makespan. To be more specific, makespan refers to the time consumed in scheduling the user-requested task in an optimal manner. This is mathematically expressed as given below.

$$Ms = n * t[\text{SOT}]. \tag{22}$$

From equation (22), makespan "Ms" refers to the product of the user requested tasks in the queue "n" and the time consumed in scheduling one user requested task "t [SOT]". It is measured in terms of milliseconds (ms). The results for performance metrics of makespan are given in Table 4. The simulation results infer that DGW-SBDRmethod reduces the makespan upon comparison with DEER [1] and load balancing for FoT-Gateways [6].

Figure 5 Average makespan of various tasks given above illustrates the evaluation of performance metrics of underlined IoT-fog-cloud environment with respect to makespan. The makespan metric is consisted as most critical especially for load balancing with optimized resource allocation in fog environment. The abovementioned metric is evaluated in milliseconds. It shows that the time consumed in scheduling user-requested tasks majorly contributes to the total time of serving fog nodes. However, time consumed in scheduling userrequests is playing a major role in average response time. This is due to the forwarding of user-requested tasks towards the nearest available fog. From a performance point of view, lower values of all abovementioned makespan metrics are preferred for better service provisioning, and on the contrary higher makespan may degrade the overall performance. The makespan was found to be comparatively lesser using DGW-SBDR than [1, 6]. The reason behind the improvement was owing to the application of three different mechanisms, i.e., node update, relocation, and differential evolution function for three distinct cases, i.e., underutilization, overutilization, and migration. As a result, the makespan using the DGW-SBDR method is said to be comparatively lesser by 18% compared to [1] and 30% compared to [6], respectively.

4.2.3. Performance Analysis of Latency and Findings. Latency refers to delays that specifically happen when any user requested tasks and waits for another user to complete the task. In other words, it is the time taken by the load balancer to handle the user request. This is mathematically stated as given below.

Table 1: Storage layer description.

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

Table 2: Column fields description.

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

Table 3: Simulation results for load balancing efficiency.

Number of tasks	Load balancing efficiency (%)		
	DGW-SBDR	DEER	Load balancing for FoT-Gateways
5000	97.7	94.7	93.1
10000	96.35	93.15	91.85
15000	95.85	93	91
20000	94.25	92.75	89.95
25000	94.15	92.55	89.35
30000	93.85	92.15	89.15
35000	92.15	91	88.35
40000	91.35	90.35	86.25
45000	91	88.15	85.15
50000	90.25	86.35	83

$$L = \sum_{i=1}^{n} n * \text{Time}[LB(T_i)]. \quad (23)$$

From equation (23), latency factor "L" is measured based on the number of user-requested tasks placed in the queue "n" and the time is taken load balancer to handle the user request "Time$[LB(T_i)]$." It is measured in terms of milliseconds (ms). Table 5 given below lists the latency rate using the three methods, DGW-SBDR, DEER [1], and load balancing for FoT-Gateways [6], respectively.

Figure 6 shows latency for 50000 user-requested tasks. X denotes the number of tasks and y represents millisecond latency (ms). According to the figure, increasing the number of jobs causes the load balancer to focus on more nodes to avoid underutilization, overutilization, or migration. Adding tasks increases the delay in all three techniques. DGW-SBDR improves significantly. Because of Differential Evolution-based Grey Wolf Optimization. This model controls optimal load balancing via load balancer using alpha, beta, and delta. Differential Evolution updates scale factors from underutilization to overutilization or between

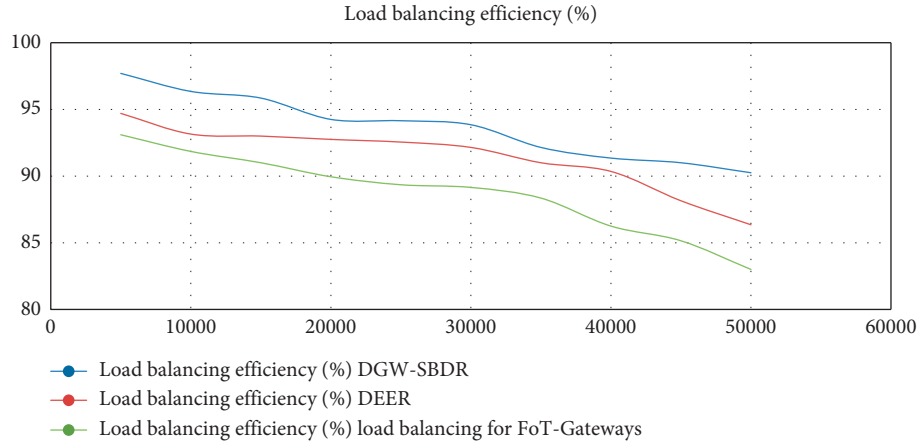

Figure 4: Average load balancing efficiency.

Table 4: Simulation results for makespan.

Number of tasks	Makespan (ms)		
	DGW-SBDR	DEER	Load balancing for FoT-Gateways
5000	12.5	15	17.5
10000	14.35	17.85	22.35
15000	16.15	21.45	25.15
20000	18.25	23.25	29
25000	21	25	31.35
30000	23.55	28.15	33
35000	25.25	31.35	35.45
40000	28.15	33.45	37
45000	30.45	35	41.35
50000	33.25	40.25	44

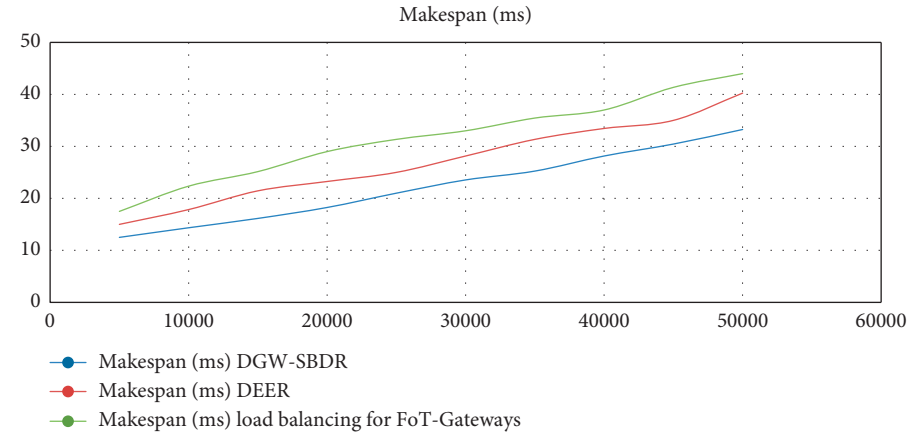

Figure 5: Average makespan of various tasks.

overutilization and migration, minimizing the likelihood of falling into local optimal policy. DGW-SBDR reduces latency by 26% compared to [1] and 34% compared to [6].

4.2.4. Performance Analysis of Energy Consumption and Findings. Consuming energy ensures effective resource allocation. It is the energy used to provide services (i.e., allocate resources) to end consumers. Energy saving leads to more effective mechanisms, reducing waste.

$$EC = n * EC(RA). \tag{24}$$

From equation (24), energy consumption "EC" is measured based on user-requested activities sitting in the queue "n" and energy consumed for resource allocation "EC (RA)" in joules "J." Table 6 presents simulation findings of

Table 5: Simulation results for latency.

Number of tasks	Latency (ms)		
	DGW-SBDR	DEER	Load balancing for FoT-Gateways
5000	5250	6750	7750
10000	5535	7135	8250
15000	5815	8245	8835
20000	6025	8295	9515
25000	6135	8355	9735
30000	6245	8515	9915
35000	6535	8925	10155
40000	6915	9015	10235
45000	7025	9125	10535
50000	7345	10355	10825

energy usage using DGW-SBDR, DEER [1], and FoT-Gateway load balancing [6].

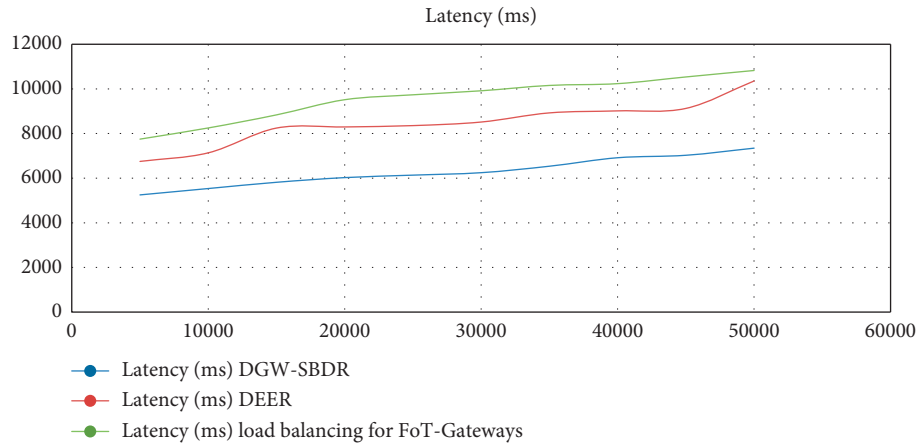

Figure 6: Average latency of various tasks.

Table 6: Simulation results for energy consumption.

Number of tasks	Energy consumption (*J*)		
	DGW-SBDR	DEER	Load balancing for FoT-Gateways
5000	25	30	40
10000	30	35	45
15000	30	35	50
20000	35	40	55
25000	40	45	55
30000	40	45	60
35000	45	50	65
40000	50	55	65
45000	55	60	70
50000	55	65	75

Figure 7 depicts resource allocation and energy usage. Energy usage is related to the number of tasks, as shown. Adding tasks increase energy utilization and vice versa. With 5000 user-requested activities, DGW-SBDR used 25 J, [1] 30 J, and [6] 40 J. DGW-SBDR consumes less energy than [1, 6] for efficient resource allocation. Stochastic Bellman Gradient Deep Reinforcement Learning-based Resource Allocation improved performance. This technique initially obtains the user-requested state space. Then, the fog created the user-requested action space. Stochastic Bellman Gradient was used to get the reward function for optimum resources. This lowered DGW-SBDR energy usage by 12% compared to [1] and 31% compared to [6].

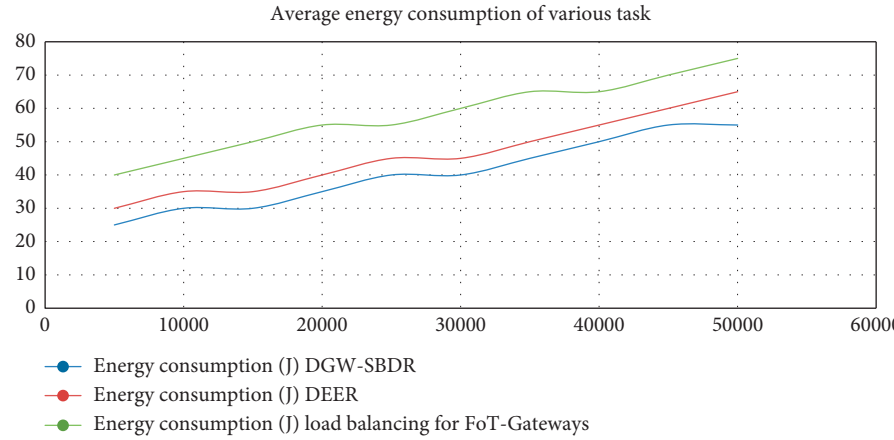

Figure 7: Average energy consumption of various tasks.

Table 7: Summary of results DGW-SBDR.

Number of tasks	Load balancing efficiency (%)	Latency (ms)	Makespan (ms)	Energy consumption (*J*)
		DGW-SBDR		
5000	97.7	5250	12.5	25
10000	96.35	5535	14.35	30
15000	95.85	5815	16.15	30
20000	94.25	6025	18.25	35
25000	94.15	6135	21	40
30000	93.85	6245	23.55	40
35000	92.15	6535	25.25	45
40000	91.35	6915	28.15	50
45000	91	7025	30.45	55
50000	90.25	7345	33.25	55
Result	Efficiency increased	Latency reduced	Time consumed in scheduling optimal manner	Energy consumption minimum

4.2.5. Summary of Results. Table 7 shows findings are measured in four categories: load balancing efficiency, Latency, makespan (the time it takes the load balancer to handle a user request), the energy needed to offer services, and the chart above compares these categories for DGW-SBDR.

The findings for load balancing effectiveness, latency (ms), makespan (ms), and energy usage are displayed in the above table (*J*). After comparing the aforementioned findings to those of the DEER algorithm and FoT-Gateways.

5. Conclusion

Cloud computing is nonideal for resource-optimized apps due to high energy consumption and latency. Fog computing uses resources near the network's edge to ensure speedy processing. Load balancing across fog nodes reduces latency, energy consumption, and ensures quick data processing due to their limited computation and storage capacities. In this study, we propose DGW-SBDR for fog load balancing and resource optimization. The load balancer is informed of user-requested activities by processing sensor-generated data. After processing, a Differential Evolution-based Grey Wolf Optimization technique is developed to accomplish overutilization via node update, and underutilization via relocation and migration. Using the Stochastic Bellman Gradient Deep Reinforcement Learning-based Resource Allocation technique, fog nodes are allocated optimum and energy-efficient resources. Simulations were used to test DGW-SBDR, DEER, and FoT-Gateways load balancing. The suggested DGW-SBD Methods beat DEER and FoT-Gateway load balancing in terms of load balancing efficiency, makespan, latency, and energy usage.

Data Availability

The dataset used to support the findings of this study can be accessed through the link Load and Transfer Test https://ast-deim.urv.cat/pc_measurement/measurement_sugarsync_load_transfer_pl.csv from https://cloudspaces.eu/results/datasets.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Supplementary Materials

The dataset used to support the findings of this study can be accessed through the link Load and Transfer Test (PlanetLab, from 2012-07-11 16:03:53 to 2012-07-17 09: 37:24) - SugarSync (http://ast-deim.urv.cat/pc_measurement/measurement_sugarsync_load_transfer_pl.csv) from http://cloudspaces.eu/results/datasets (*Supplementary Materials*)

References

[1] Z. A. AneesUrRehman, M. AlaaAlaanzy, O. Mohamed, A. I. Umar, and J. Ahmad, *Dynamic Energy Efficient Resource Allocation Strategy for Load Balancing inFog Environment*, IEEE Access, Piscataway, NY, USA, 2020.

[2] J. Hu and Q. Yang, "Dynamic energy—efficient resource allocation in wireless powered communication network," *Wireless Networks*, vol. 25, no. 6, pp. 3005–3018, 2019.

[3] A. Forestiero, "Heuristic recommendation technique in Internet of Things featuring swarm intelligence approach," *Expert Systems with Applications*, vol. 187, 2022.

[4] S. Mirjalili, S. M. Mirjalili, A. Lewis, and G. W. Optimizer, "Advances in engineering software," *Grey Wolf Optimizer*, vol. 69, pp. 46–61, 2014.

[5] F. Gul, I. Mir, L. Abualigah, P. Sumari, and A. Forestiero, "A consolidated review of path planning and optimization techniques: technical perspectives and future directions," *Electronics*, vol. 10, no. 18, p. 2250, 2021.

[6] E. Batista, G. Figueiredo, and C. Prazeres, "Load balancing between fog and cloud in fog of things based platformsthrough software-defined networking," *Journal of King Saud University –Computer and Information Sciences*, Elsevier, Amsterdam, Netherlands, 2021.

[7] M. MohdShahriarMaswood, Md. RahinurRahman, and A. G. Alharbi, *A Novel Strategy to Achieve Bandwidth Cost-Reduction and Load Balancing in aCooperative Three-Layer Fog-Cloud Computing Environment*, IEEE Access, Piscataway, NY, USA, 2020.

[8] S. Nabi, M. Ibrahim, and J. M. Jimenez, *DRALBA: Dynamic and Resource AwareLoad Balanced Scheduling Approachfor Cloud Computing*, IEEE Access, Piscataway, NY, USA, 2021.

[9] M. Junaid, A. Sohail, O. Khalid, I. A. Khan, S. H. Syed, and N. Ejaz, *Modeling an Optimized Approach for LoadBalancing in Cloud*, IEEE Access, Piscataway, NY, USA, 2020.

[10] M. Fatma, *Talaat, Effective Prediction and Resource Allocationmethod (EPRAM) in Fog Computing Environmentfor Smart Healthcare System, Multimedia Tools and Applications*, Springer, Berlin, Germany, 2022.

[11] M. Iyapparaja, M. S. Kumar, S. Siva Rama Krishnan, and C. Lal Chowdhary, "Efficient resource allocation in fog computing using QTCS model," 2021, https://www.techscience.com/cmc/v70n2/44610.

[12] H. Tran-Dang, S. Bhardwaj, T. Rahim, A. Musaddiq, and D. S. Kim, "Reinforcement learning based ResourceManagement for fog computing environment:literature review, challenges, and open issues," *Journal of Communications and Networks*, vol. 24, no. 1, pp. 83–98, 2022.

[13] S. Mishra, M. N. Sahoo, J. J. SambitBakshi, and P. C. Rodrigues, *Dynamic Resource Allocation in Fog-Cloud HybridSystems Using Multi-Criteria AHP Techniques*, IEEE, Piscataway, NY, USA, 2020.

[14] C. Yi, S. Huang, and J. Cai, *Joint Resource Allocation for Device-To-Device Communication Assisted Fog Computing*, IEEE Transactions on Mobile Computing, Piscataway, NY, USA, 2019.

[15] F. Hosseinpour, N. Ahmad, and S. Virtanen, *A Resource Management Model forDistributed Multi-Task Applicationsin Fog Computing Networks*, IEEE Access, Piscataway, NY, USA, 2021.

[16] M. AbdElaziz, L. Abualigah, and I. Attiya, "Advanced optimization technique for scheduling IoT tasks in cloud-fog-computing environments," *Future Generation Computer Systems*, Elsevier, Amsterdam, Netherlands, 2021.

[17] Z. Chang, L. Liu, XijuanGuo, and S. Quan, *Dynamic Resource Allocation and ComputationOffloading for IoT Fog Computing System*, IEEE Transactions on Industrial Informatics, Piscataway, NY, USA, 2020.

[18] Q. Ren, K. Liu, and L. Zhang, *Multi-objective Optimization for Task Offloading Based on Network Calculusin Fog Environments, Digital Communications and Networks*, Elsevier, Amsterdam, Netherlands, 2021.

[19] K. Gai, X. Qin, and L. Zhu, "An energy-aware high performance TaskAllocation strategy in heterogeneous FogComputing environments," *IEEE Transactions on Computers*, vol. 70, no. 4, pp. 626–639, 2021.

[20] M. Keerthika, N. Sankar Rama, M. Rajkumar, M. V. Manivannan, and S. Monish, "To optimize the multi accesses download time using schedulingapproach in fog computing," *Materials Today: Proceedings*, Elsevier, Amsterdam, Netherlands, 2020.

[21] X. Liu, B. Lai, L. Gou, C. Lin, and Mu Zhou, *Joint Resource Optimization for UAV-EnabledMultichannel Internet of Things Based onIntelligent Fog Computing*, IEEE Transactions on Network Science and Engineering, Piscataway, NY, USA, 2020.

[22] M. Abdel-Basset, R. M. NourMoustafa, O. M. Elkomy, and M. Abouhawwash, *Multi-Objective Task Scheduling Approachfor Fog Computing*, IEEE Access, Piscataway, NY, USA, Sep 2021.

[23] Z. Movahedi, B. Defude, and A. M. Hosseininia, "An efficient population-basedmulti-objective task scheduling approach infog computing systems," *Journal of Cloud Computing*, vol. 10, no. 1, p. 53, 2021.

[24] Simona-VasilicaOprea and A. Bâra, *Edge and Fog Computing Using IoT for Direct Load Optimization Andcontrol with Flexibility Services for Citizen Energy Communities, Knowledge-Based Systems*, Elsevier, Amsterdam, Netherlands, 2021.

[25] RaúlGracia-Tinedo, M. Sánchez-Artigas, A. Moreno-Martínez, C. Cotes, and P. García-López, "Actively measuring personal cloud storage," in *Proceedings of the 6th IEEEInternational Conference on Cloud Computing*, pp. 301–308, 2013, https://www.thecloudcomputing.org/2013/.