ISSN 1330-3651 (Print), ISSN 1848-6339 (Online)
Received: 18 July 2025; Accepted: 24 November 2025

https://doi.org/10.17559/TV-20250718002842
Original scientific paper

Sustainable Energy Modelling in Cloud-Based Healthcare Systems Using Hybrid Grey Wolf Firefly Optimization

P. Manikanda PRABU*, V. R. Sarma DHULIPALA

Abstract: The Cloud computing is a technology breakthrough that delivers utility-based services to a variety of consumers. In recent times, the scalability and economy of cloud computing have been recently accepted and used in the healthcare industry. It allows the healthcare sector to provide a top platform for the consumers to have access to the best medical treatments in the market. The impact of cloud computing on health care operations with respect to the rule of supplying services, cooperation, building operational models, and provision of end user's services is great and permanent. Electronic Health Record (EHR) applications have evolved leading to higher demand for the computing resources of data centers. The most interesting uses of electronic health record is the instant and instant access to healthcare data for consumers and patients. We aim to accomplish a general performance and usage of cloud resources through their high efficiency in this work for effective energy and load balancing. Specifically, the suggested method, which officially uses the name HWGFF, is a hybrid optimization of Grey Wolf Firefly algorithm to achieve the optimal energy usage and load balancing in transmission. The presented approach is an under a cloud computing context to maximize resource utilization while minimizing energy usage and expenditures. Although the result of the technique we suggested is not very convenient, there is at least a more sensible answer. It is found experimentally according to the approach that the suggested is somewhat more efficient than the currently used load balancing techniques.

Keywords: cloud computing; IoHT; load balancing; medical service energy optimization; optimization in task allocation

1 INTRODUCTION

Healthcare is no exception with the revolution that is swiftly occurring in IT, thanks to the Internet of Things (IoT) [1-3]. Chronic illness management and mobile health and remote patient monitoring require IoT device. But many applications require real time processing of data, so even a lag in handling the data is not tolerated. Fog computing works as a good combination in the middle of sensors, such as cloud computing, to improve system efficiency by more effective collecting and processing data [4].

The enormous data produced by the IoT devices can be processed and stored by the offered processing and storage capacities under cloud computing. However, it is difficult to achieve minimal latency and good bandwidth usage with the help of conventional cloud computing for real time applications such as in healthcare. However, extending cloud capacities to the network edge using fog computing minimizes these difficulties by making computer resources available in the network edge, closer to IoT devices [5]. In healthcare, in particular, it is all the more important to process data quickly, as it could be part of the patient care process.

Scheduling decisions are facilitated by heuristic algorithms which use elementary rules and measures for that end. Often these methods are computationally efficient, and will provide almost optimal solutions in many examples [6]. The first one is the First-Come, First-Served (FCFS) scheduler. They simply execute the tasks in the order that they have arrived, as the name suggests. This is an easy approach; however, it may not be useful for real time applications with strict deadline constraints. Shortest Job First (SJF) algorithm is such an adaptive process that executes tasks with minimal processing time. This algorithm can decrease the average waiting time on the shortest coming task, but longer tasks can starve for longer. It utilizes activities classification and virtual machines categorization (TCVC) along with the MAX\MIN scheduling algorithm in order to schedule or assign jobs in high priority. To improve performance [7] tasks are divided into high, medium and low relevance.

Mathematical models and search methodologies are used for finding the optimal or near optimal schedule by the optimization algorithms [8]. Thus, these algorithms produce better than the heuristic based algorithms do, but they are generally much more computationally expensive. The Genetic Algorithm (GA) is a population based search method that uses natural selection and genetics as the search domain, to find optimum solution. Many objectives can be achieved minimizing work time, energy, and cost and by doing so, work scheduling can be optimized.

The Reinforcement Learning, an innovative offloading methodology, chooses the optimal node to acquire a balanced workload distribution between the nodes using a Deep Q Network (DQN) method, which is one of the types of deep reinforcement learning methods [9]. The dynamic scheduling of the fog cloud computing framework is achieved through hybrid Deep Belief Network (DBN) that is a combination of Grey Wolf Optimizer (GWO) and Lion Algorithm (LA). Specifically, the DBN has shown improved performance as related with energy expenditures, processor expenses, and communication prices [10].

2 RELATED WORKS

In this research [11], novel offloading is presented to be used in Internet of Things IoT applications using a three tier fog computing framework. The foundation stone of this design is the IoT layer, which is saturated with intelligent devices generating a large amount of activities. They are diverse jobs with varying attributes including data volume, computation resources needs, communication overhead, and time critical latency requirements. To dynamically choose the correct node for task distribution, the study uses Deep Q-Network (DQN), a kind of deep reinforcement learning. The purpose of this DQN-based methodology is to equitably distribute the workload among the available

fog nodes [12]. Experimental results demonstrate effectiveness of the proposed offloading technique.

This paper [13] proposes Mobility aware Secure Computation Offloading (MSCO), a novel Mobility aware Secure Computation Offloading (MSCO) technique that improves the performance and secure applications latency in fog computing environment for IoT. The algorithm combines genetic algorithms and particle swarm optimization and is referred to as hybrid. According to the research that presents the experimental results, the MSCO mechanism offers a considerable improvement in comparison to existing methods, as the average cost reduction is by up to 11% in the case of total (both in terms of latency and energy consumption) cost.

In this research [14], an optimization framework that leads to energy minimization in several layered Internet of Things (IoT) architectures, provably optimizing most of the aforementioned situations, is presented. The framework resides in a stratified edge-fog-cloud architecture for a network so we can distribute compute across multiple tiers of the network. This energy aware optimization is examined in many cases of use, such as smart grids, driverless vehicles and e-health applications. Specific ideas are demonstrated through CPLEX simulations, using which it is able to quantify the meaningfulness of designing procedures to pick nodes which will maximize energy use, application desires and present network confines in such difficult designs.

Research [15] concludes that the cloud fog collaboration environment (CFC) is an appropriate choice for real time IoT data sensing and workflow applications in the cases of high internet access. Workflow scheduling is defined as the efficient distribution of interrelated tasks to available computational resources of heterogeneous computing systems (HCS) and this is the main focus of the work. This work presents a new Hybrid-metaheuristics Multi-objective Workflow Scheduling Algorithm (HMWSA) that solves the problem of workflow scheduling while considering that workflow scheduling strategies that decrease latency are important in HCS [16]. Following the CFC setting, this method integrates Harmony Search (HS) and Genetic Algorithm (GA) in carefully couched manner for the purpose of enhancing workflow scheduling. An innovative hybrid methodology consisting of a Tuna Swarm-optimized Bacterial Foraging Optimization (TBFO) algorithm combined with a Deep Neural Network (DNN) is presented in paper [17].

3 MATERIALS AND METHODS

To load balance inside a cloud environment, we suggest to implement HGWFF. Fruit flies are used as an agent in this method to bring about a state of equilibrium of the workloads among virtual computers. There are two search stages enacted by Firefly Optimization Algorithm (FOA), the olfactory and the visual [18]. Virtual machines load themselves with tasks, either overloaded or under utilized and multiply those tasks to other virtual machine which is neither overloaded nor underutilized [19]. In HGWFF we rely on the search phases to find a good virtual machine for the task to assign to after deletion. Establish the multi objective function and the load balancing model can be used if such function limits are obeyed [20]. The method employs a threshold value to ascertain the load of a virtual computer. The planned work diagram is illustrated in Fig. 1. The division of task management of the proposed HGWFF has four segments. The system rescinds the task when a virtual machine is overcrowded and then reassigns it to the virtual machine designated for it based on the latest deadline that it has to fulfil. To increase the datacenters performance, we will choose the virtual machine with the least deadline task [21]. For example, olfactory based search locates accessible locations or virtual machines, whereas visual based search finds the most appropriate location or virtual machine for the migration of the task of the overloaded source.

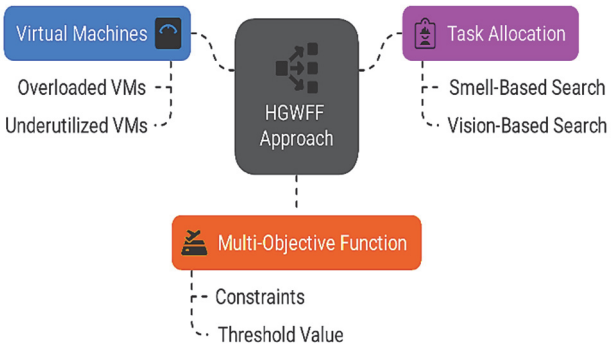


Figure 1 The proposed workflow of HGWFF algorithm

3.1 System Model

For an example, we will be considering the IoT fog cloud architecture where a IoT layer with a collection of distributed IoT devices and end users [22], a fog layer with various fog nodes, a cloud layer with a cloud DC are present. The fog layer may be directly concerned with IoT devices in terms of producing a sequence of requests and transmitting them to a fog node located within the same operational domains of the IoT devices [23]. In a functional area, also called the fog instance (FI), will be deployed one or more IoT-fog-cloud applications [24]. The fog is used to minimize load on central cloud and to reduce service latency for near devices. The majority of time sensitive requests of IoT devices can be managed by fog nodes for the most part and rest to be forwarded to cloud for processing [25].

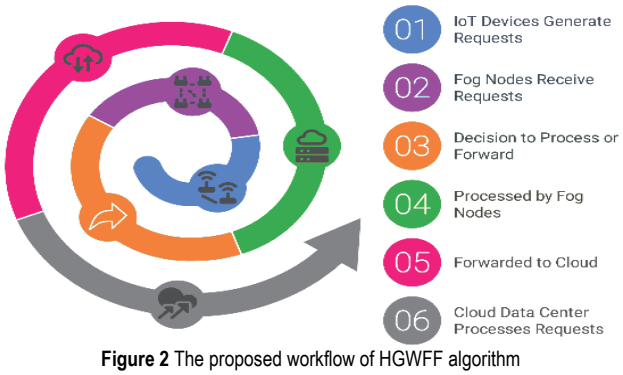


Figure 2 The proposed workflow of HGWFF algorithm

Based on the fact that the fog nodes can aggregate with each other through LANs, we refer to them as Fog Instances, endowed with numerous resources. Thus, we can analyse the Erlang delay systems [26] for each of the fog nodes and thus we treat each of them as such an $M/M/n$ queueing system. The fog infrastructure has k fog nodes and nk servers (n_k are online for supporting the IoT devices), the fog nodes need to work with the cloud data

center [27]. Therefore, we assume that all requests from end user will arrive at fog layer. Processing of a fog node is a prerequisite to then determining whether to process or send a request. The following stages are executed by each initiation process of fog nodes as illustrated in Fig. 2.

3.2 Model of Fog Node Processing

Let us consider d IoT devices transmit the request to f fog nodes via wireless. This request packet size is P bytes. Then the rate is defined as,

$$1/R_f = 1/[PW(1+(h_d f P_t x^{(df)})/(WN_0))] \quad (1)$$

where channel gain is h_{df} and P_{tx}^{df} is transmission power [23]. W is channel bandwidth with noise power of N_0. The power consumption of fog node f depends on hardware architecture parameters of κ & F^f. F is frequency and s is clock speed derived from frequency F. I_p^* is idle power of fog server when it is static. The mean power is represented as,

$$P_{\text{fog}} = \kappa s^{\phi} + I_p^* \quad (2)$$

An IoT device sends the results of the fog node's processing to the cloud once the fog node is finished. The incoming data are smaller than the magnitude of the calculation in more than one case [28]. A fog node does not consider time and energy spent in relaying query results whereas for transmission delay and energy usage.

3.3 Cloud Processing Model

Working with the data center hosts which are the cloud, we model the cloud as an $M = G = 1$ system in which the queue time of requests can be disregarded and the hosts are in possession of unlimited servers with unlimited resources and capacity, and evaluate the cloud requests' batch flow in the Poisson process [29]. So we use the notation T_c to refer to the time taken to process a request in cloud's data centre and it is given by,

$$T_c = \frac{1}{\mu_{\text{cld}}} \quad (3)$$

Furthermore, the results of the cloud processing will be returned to the fog after processing is completed [30]. The steps that the equipment must perform to process the outcomes of the requests are not counted.

3.4 Proposed HGWFF Algorithm

Eventually, the intention of this work is to bring load equilibration among virtual machines of a cloud datacenter through a hybrid Grey Wolf and Fruit Fly (HGWFF) approach for optimization. The major disadvantage of fruit fly optimization is that it is not very precise in the optimization and is quite sensitive to local optimum. The main objective of HGWFF was to overcome the limitations of Fruitfly technique which first developed. The plan suggested contains such an approach, which consists of two elements. In the start off portions of the fly over area (FOA) procedure, every single swarm of flies shall distribute in diverse directions to see to it that distribution is kept fixed. In the second stage, Grey wolf optimization (GWO) is used to amend the current locations and solutions [31], due to its exploration and exploitation potentials. The FOA is also applied to solve the early convergence problem associated with FOA. That is, it brings the convergence rate and optimization accuracy in some improved ways.

Consider virtual machines as $V(v_1, v_2, ..., v_Z)$, real machines as $M(m_1, m_2, m_3, ..., m_Z)$ and number of tasks are $T(t_1, t_2, t_3, ..., t_S)$. The task variable holds the information about arrival time a_i, length of task l_i, deadline of the task d_i, task finishing time f_i. On every slot the task is assigned to VM via cloud intermediator. The processing time of all tasks is denoted as

$$T_n = \sum_{i=1}^{k} T_{ij}, \quad j = 1, \ldots, n \quad (4)$$

The capacity of virtual machine is defined as

$$C_{vi} = P_{(e-\text{count})} P_{(e-\text{mips})} B_{vi} \quad (5)$$

where $P_{(e-\text{count})}$ is processing elements's processor count, B_{vi} is the bandwidth of virtual machine i, $P_{e-\text{mips}}$ is mips of all processors. The load processing by each VM is given by

$$L_V(t) = \frac{N(T,t)}{SR(V_j,t)} \quad (6)$$

where $N(T,t)$ is number of tasks assigned, $SR(V_j,t)$ is rate of service for specific VM at time t. Hence, load of all virtual machines is:

$$L_A = \sum_{k=1}^{Z} L_V(k) \quad (7)$$

Processing time of virtual machine

$$T_v = \frac{L_{V(k)}}{C_{vi}} \quad (8)$$

Processing time of all virtual machines

$$T_A = \frac{\text{Load of all } k}{\text{Capacity of all } k} \quad (9)$$

Execution time of Task T:

$$E(T) = \frac{\zeta_T}{c(V_j)} \quad (10)$$

where task size is ζ_T, T and CPU performance is derived in $c\left(V_j\right)$. The finishing time fn_{ik} of the task T with start time of st_{ik} is given by

$$fn_{ik} = st_{ik} + E(T) \tag{11}$$

To ensure one task is assigned to only one VM, one decision variable is applied here as β_{ij} and it is defined as

$$\beta_{ij} = \begin{cases} 1, & \text{if VM assigned and } f_i < d_i \\ 0, & \text{if } f_i > d_i \end{cases} \tag{12}$$

A collection of all virtual machines Vj and a task Tk at random are assigned to every fruit fly to start the developing of a solution. First, the method checks to see if the loads on each virtual machine fit into a specified value of threshold. If a virtual computer exceeds its capacity, the task that is being performed will be stopped [31]. The virtual computer has to be found by each solitary fruit fly so the eliminated labor can be distributed. The first of all the virtual machines to be eligible for job transfer is the manager for food. The approach will repeat until a given datacenter state is that in which no virtual machine has unbalanced load [32]. The workflow of the proposed HGWFF is shown in Fig. 3 below. The algorithm of HGWFF is given in Algorithm 1 below.

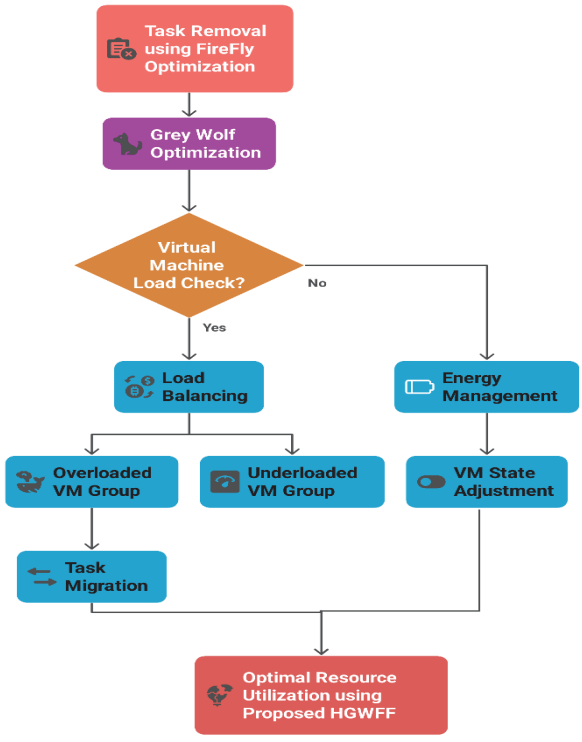


Figure 3 The proposed workflow of HGWFF algorithm.

Algorithm 1: HGWFF Optimization-Based Load Balancing

Inputs:

- Number of iterations: I
- Number of wolf search agents: S
- Search space dimension: k

Initialization:

- Set the control coefficient $a = 2$
- Randomly generate coefficients r_1 and r_2
- Initialize the positions of all grey wolf agents
- Compute the fitness of all wolf agents
- Identify:
  - α wolf → leading wolf
  - β wolf → superior wolf
  - δ wolf → inferior wolf

Begin Optimization Loop

For each iteration $i = 1$ to I:

For each search agent $s = 1$ to S:

Update the position of each agent

Compute the updated positions using the current α, β, and δ wolf information.

Average the updated candidate positions to obtain the new position of the search agent.

Compute distances

Determine the distance of the search agent from α, β, and δ wolves.

Compute coefficient vectors

Update the control vectors based on random numbers and the current value of a.

End for

Update the coefficient 'a'

Reduce the value of a linearly as the iteration progresses.

Recalculate the fitness of all agents

Evaluate the new fitness of each wolf.

Update α, β, and δ wolves accordingly.

If the α wolf is modified:

Update load-adaptive balancing coefficient

Compute the adaptive coefficient l based on the current iteration.

If a random number is greater than l:

Update firefly positions

Recalculate the firefly movement probability based on the distances between the search agent and other wolf agents.

End If

End If

End For

4 PERFORMANCE ANALYSIS

Two independent experiments are given in the next section. The first experiment compares the suggested HGWFF with past optimizations. The experiment employs six different benchmark optimization techniques. For our experiment, we pick a size of 50 flies and repeat 100. The second experiment investigates if HGWFF balances loads in a cloud environment for EHR programs, using the CloudSim tool, a good tool that simulates and models the situations in cloud computing [33]. We have evaluated the effectiveness of the proposed approach based on the results of the simulations. We assess the various load assignments in a scenario comprising 50 nodes with homogenous OBUs, task requests ranging from 250 to 1000, 25 fog servers. The processing energy of fog nodes exhibits a 52% increase in energy efficiency relative to the cloud and OBU nodes is 90% more efficient than that of cloud servers. Subsequently, Tab. 1 provides the configuration details of the experiment in processing capability and power network of components throughout the various tiers of the system.

To simulate the proposed method is done using multi objective functions developed inside the given constraints. Then, we compare the cloud load coverage before and after load balancing takes place. Our proposed method is energy

and cost effective and focuses not only on load balancing issues particular to data centers but also on the efforts of the datacenter data regarding the usage of the energy.

Table 1 Simulation configurations

Network Components	Capacity	Energy consumption
Cloud switch	320 Gbps	3.8 kW [23]
Cloud router	660 Gbps	5.1 kW [24]
Optical switch	125 Gbps	64 kW [25]
Core router	13 Tbps	14 kW [26]
Edge router	200 Gbps	4.2 kW [27]
Aggregation switch	160 Gbps	3.8 kW [28]
Cloud Server	4 GHz	300 W [29]
OBU processor	1 GHz	7 W [30]
Fog server	2.7 GHz	95 W [31]
Access point	2 Gbps	7.5 W [32]
Network Wifi	75 Mbps	200 mW [33]

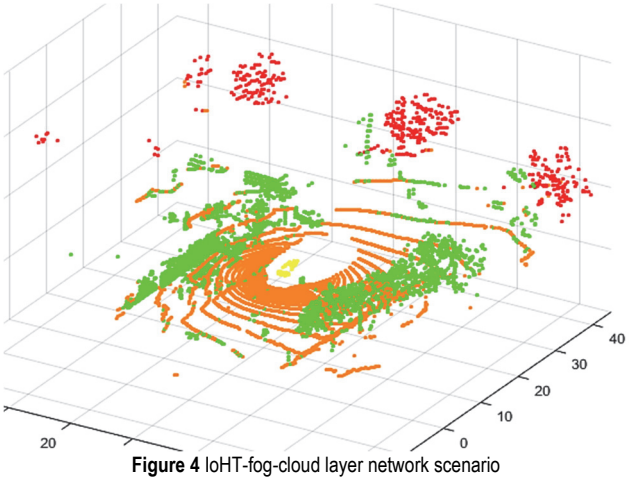
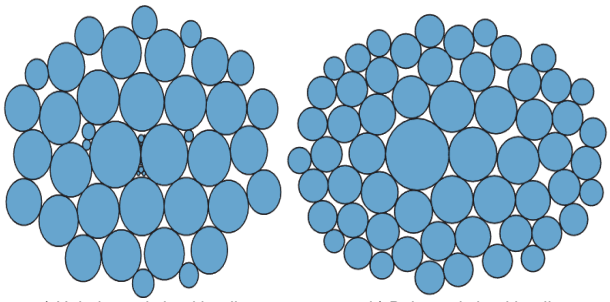


Figure 4 IoHT-fog-cloud layer network scenario

Fig. 4 illustrates the topological view of the scenario considered for our implementation which has three layers in the entire network model. The bottom layer indicated by orange color is IoHT nodes in Fig. 5. The density of nodes in IoHT layer is always higher than other two layers of Fog and Cloud. The second layer of Fog is depicted as green points as shown in Fig. 5. The requests are processed and propagated by the Fog layer from IoHT to Cloud. Finally the cloud points are marked as red points which collect the information from the fog nodes. In Fig. 5, we show the cloud coverage before and after using the proposed HGWFF based load balancing methodology. Each circle in these two figures is denoting the number of virtual machines it is handling in each cloud point.



a) Unbalanced cloud loading b) Balanced cloud loading

Figure 5 Cloud coverage before and after load balancing using the proposed HGWFF

Fig. 6 illustrates the performance of energy consumption as number of virtual machines increasing for each layer in the network using the proposed HGWFF. In IoHT layer, the energy consumption depends on the user devices. It cannot be imbalanced or reduced due to the reason of it is user end. After it enters into Fog and Cloud layer, the energy consumption is reduced and balanced using our proposed HGWFF. Energy consumption of IoHT layer ranges from 0 to 5.3 KWh. Fog layer is 0.1-2.01 KWh and cloud layer is 0.5-1 KWh. Hence the energy deviation around all nodes in cloud is 0.5 KWh which is 75% reduced than Fog layer and 99% than IoHT layer.

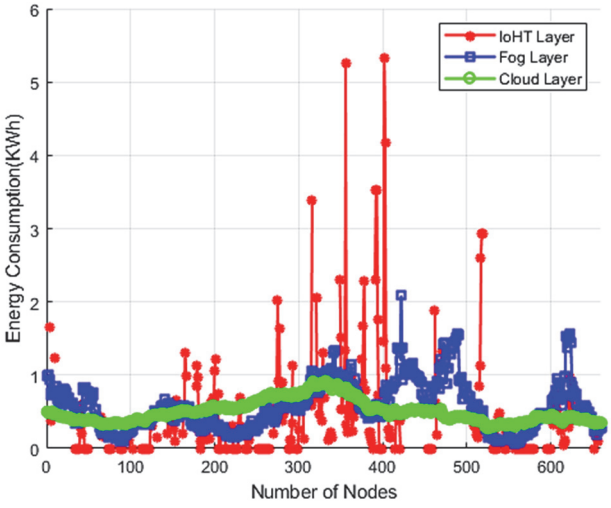


Figure 6 Energy consumption of each layer in proposed HGWFF

Fig. 7 depicts the performance of task allocation as the number of virtual machines is increasing from 5 to 25. When the processor called as virtual machines is increasing in the network, the number of task allocation to Fog nodes is increasing.

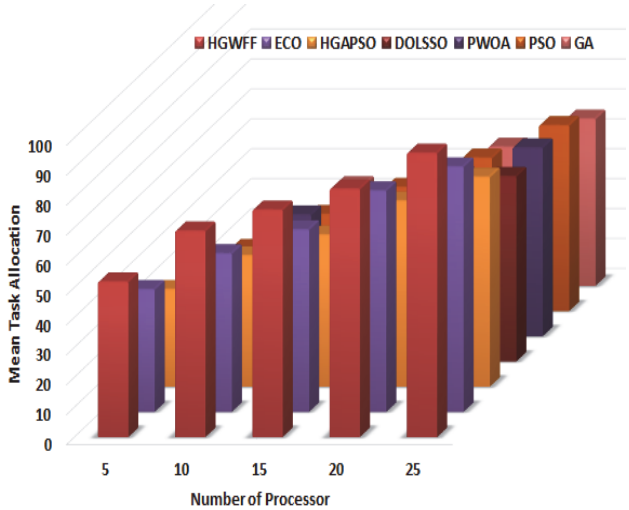


Figure 7 Number of task allocated performance

In Fig. 7, we compared our proposed HGWFF with earlier implementations of different optimization algorithms of ECO, HGAPSO, DOLSSO, PWOA, PSO, GA. The high number of tasks is allocated in our proposed HGWFF at 25 processors of 95 and for the same processor it is 82, 70, 61, 63, 62, 55 achieved by ECO, HGAPSO, DOLSSO, PWOA, PSO, GA methods respectively. Therefore, the proposed HGWFF is 14 percent very effective in the process of task allocation than the established ECO approach.

The performance of energy consumption as number of fog nodes is increasing is shown in Fig. 8. The average network energy consumed for each node scenario is measured and plotted in Fig. 8. The maximum energy utilization is not more than 5 for our proposed HGWFF.

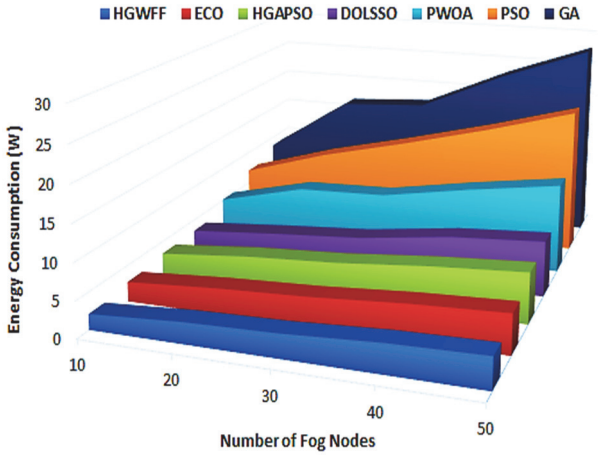


figure 8 number of task allocated performance

Fig. 9 shows the performance of transmission cost for five different Fog Scenarios of increasing loads in the network. In Each Fog Scenarios the ratio of Fog nodes and cloud points is varied to increase the transmission rate to 580 bps/Hz in our proposed HGWFF and it is 50% higher than existing method of ECO. In Tab. 2, the time performance of proposed HGWFF based optimization for load balancing is compared with existing optimizations methods of ECO, HGAPSO, DOLSSO, PWOA, PSO, GA. For time performance analysis, response time, awaiting time, process time are evaluated.

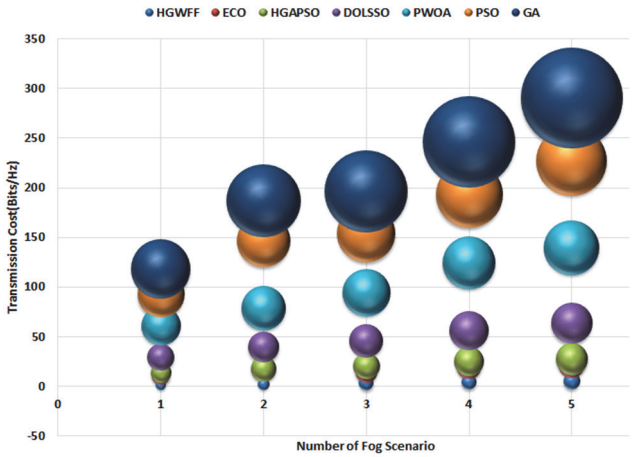


Figure 9 Performance of transmission cost of proposed HGWFF

Table 2 Time performance comparison of proposed HGWFF with existing optimization methods

Optimization Methods	Response Time / s	Awaiting Time / s	Process Time / s
GA	0.9340	0.7060	0.9502
PSO	0.8491	0.6555	0.8235
PWOA	0.7577	0.5769	0.6948
DOLSSO	0.7431	0.3712	0.4387
HGAPSO	0.6787	0.2971	0.3816
ECO	0.3922	0.1462	0.3171
Proposed HGWFF	0.0357	0.0818	0.1344

Fig. 10 depicts the performance of latency using proposed optimization HGWFF and existing optimization methodologies. When the number of tasks increases the latency will automatically increase as shown in figure 10 for all methods. But compared to other optimization, our proposed HGWFF achieves the low latency of 15.4034 ms and 575.2088 ms with Tasks of 100 and 1000 respectively.

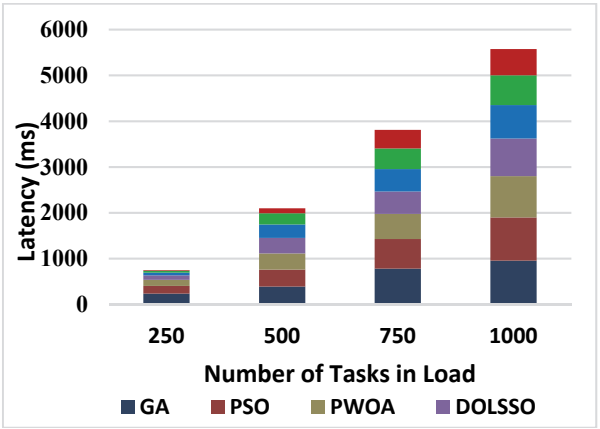


Figure 10 Latency performance of proposed HGWFF and existing optimization methods

The analysis of load balancing time is illustrated in Fig. 11.

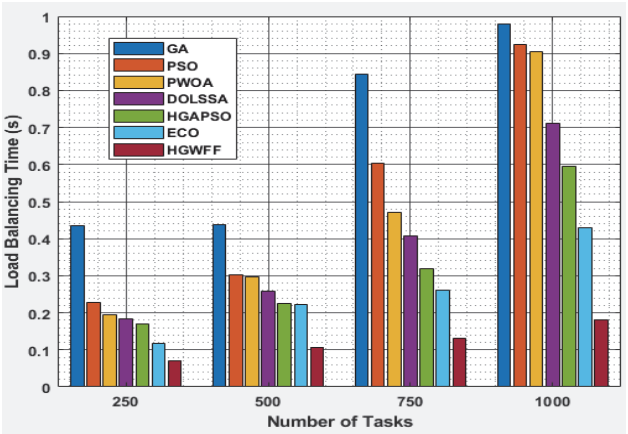


Figure 11 Load balancing time comparison of proposed HGWFF and existing optimization methods

With very fast convergence, the proposed optimization of HGWFF takes 0.0711 s at 100 task and 0.1810 s at 1000 tasks for load balancing between the number of VMs. HGWFF saves maximum of 84% of load balancing time compared to conventional GA and minimum of 58% than recent existing optimization of ECO.

In Fig. 12, the throughput comparison is performed for proposed optimization based load balancing and existing methods with the help of pie chart representation. As shown in Fig. 13, the prediction of throughput is evaluated with four conditions of 250,500, 750 and 1000 tasks variations. The HGWFF achieved the highest percentage of throughput in each case compared to other existing methods. The exact throughput represented in Mbps for each method is given in Tab. 3.

In Fig. 14, computational complexity and efficiency of each optimization method along with our proposed HGWFF is shown. Computational complexity is at its highest when dealing with complicated data, such as

heterogeneous data in communication systems that make networks perform better, which leads to more requests. The challenge of determining the cost of messages transmission, data sharing and schedules offloading in edge cloud computing.The complexity of proposed HGWFF is 9% and it is 68% less than GA method.

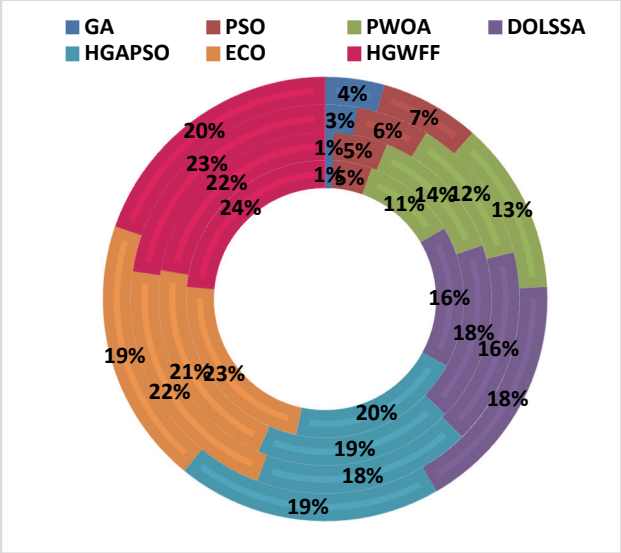


Figure 12 Pie chart comparison for throughput analysis of proposed HGWFF and existing optimization methods

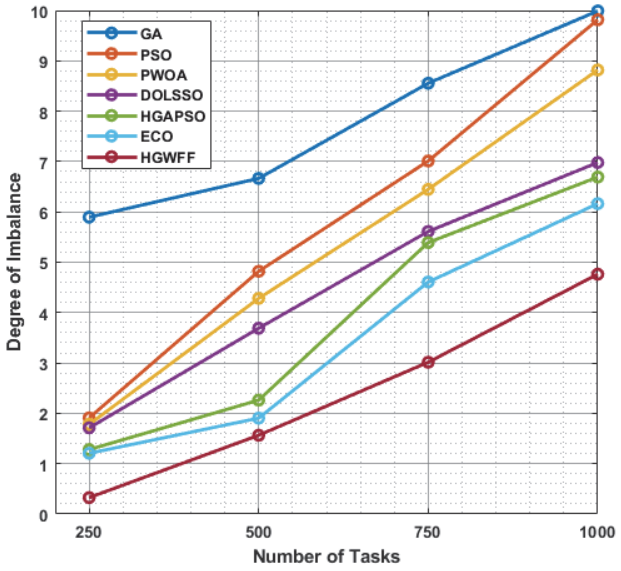


Figure 13 Degree of load imbalance perfomance comparison

Table 3 Throughput comparison of existing optimization methods with proposed HGWFF

Optimization Methods	Throughput / Mbps			
	Number of Tasks			
	250	**500**	**750**	**1000**
GA	2.8674	3.0541	10.6762	19.7810
PSO	13.6553	18.2922	23.9932	33.4163
PWOA	33.5357	47.9922	49.4174	57.6722
DOLSSO	50.0022	60.9867	65.3757	80.5489
HGAPSO	61.7666	67.9728	72.1227	88.6512
ECO	69.8746	74.4074	85.9442	89.0923
Proposed HGWFF	71.5037	80.9052	90.3721	95.4722

The effectiveness of the total execution of certain applications has significant bearing in the design of the algorithm in optimization. Proposed HGWFF attains the efficiency of 97.95% and it is 16% higher than the conventional ECO optimization.

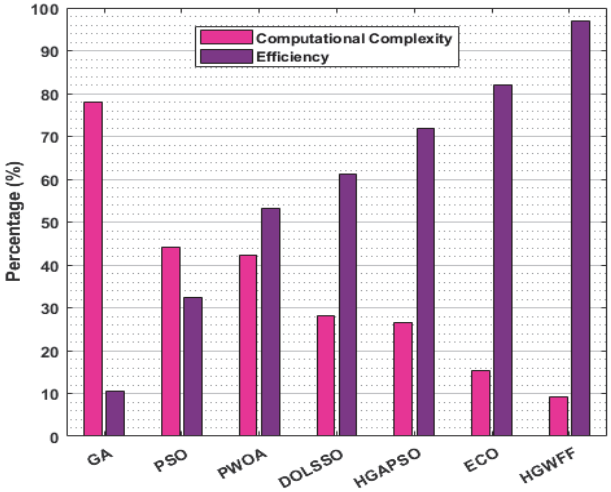


Figure 14 Percentage of computational complexity and efficiency of proposed and existing optimization

In Fig. 15, the correlation map of our four objectives attained using our proposed HGWFF is illustrated. In a correlation matrix, the diagonal column represents each objective's correlation with itself, which is always perfectly correlated . The description highlights a trade-off in network performance where increasing the Transmission Rate leads to reduced Energy and Load (negative correlation) but results in higher Latency (positive correlation) as shown in Fig. 15.

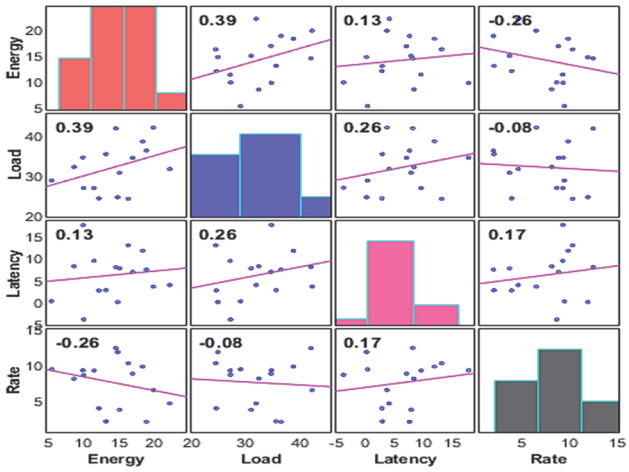


Figure 15 Correlation of metrics used in the optimization of proposed HGWFF

In the cloud computing context, HGWFF is to maximize resource use with minimal energy spending and cost. Our suggested approach leads to a result that is superior. Findings of experimental results show that the proposed method is better than the current load balancing schemes in the area of efficiency.

5 CONCLUSION

This research proposes a HGWFF multi-objective hybrid Grey Wolf and Fruitfly (HGWFF) optimization strategy for cloud computing energy efficiency and load balancing. We tested twice. The HGWFF is compared to traditional prior methodologies for several energy

economy optimization tactics in fog cloud computing in the first trials. In the second experiment, the recommended HGWFF approach is tested in several scenarios. The suggested HGWFF solves the restrictions in the present FireFly Optimization algorithm (FFOA) and yields an efficient optimal solution for virtual machine workload balancing. HGWFF uses Grey Wolf Optimization (GWO) for local search to increase convergence rate and datacenter performance. The convergence of HGWFF is better than the existing techniques. The proposed strategy is optimal in terms of workload allocation since changing the threshold values make latency, energy usage, and data centre costs minimal. A sleeping plan reduces energy consumption in the HGWFF. The proposed HGWFF was more energy efficient by 22%, 39%, 44.7%, 66.03%, 78.6%, and 85% from the conventional methods such as ECO, HGAPSO, DOLSSO, PWOA, PSO, and GA. HGWFF-based load balancing improves performance and scalability by encouraging energy-efficient resource use. Optimizing load distribution can minimize power use in healthcare data centers by concentrating activities on fewer servers and turning idle computers off. This suggested HGWFF technique meets sustainability goals while scaling resources fast. Our approach excels. We hope to strengthen our study on deep learning-based environment-dependent resource management for real-time applications.

6 REFERENCES

[1] Jeyaseelan, W. R. S., Krishnan, R., Arunkumar, M., & Alagarsamy, P. (2024). Efficient intelligent smart ambulance transportation system using Internet of Things. *Tehnički vjesnik - Technical Gazette*, *31*(1), 171-177. https://doi.org/10.17559/TV-20230726000829

[2] Davlatov, S., Zayniyev, A., Zokirov, J., Temirova, M., Uljaeva, S., Khudayberganov, K., Matkarimov, I., & Van Truong, C. (2026). Integration of Digital Twin Technology and Industry 4.0 Principles for Real-Time Structural Health Monitoring in Smart Manufacturing Facilities. *International Journal of Industrial Engineering and Management*, *17*(1), 1-20. https://doi.org/10.24867/IJIEM-400

[3] Zhang, H. K., Yang, S., Zheng, Q. M., & Liang, H. R. (2025). Optimizing emergency home healthcare scheduling with improved quantum-behaved particle swarm optimization. *Advances in Production Engineering & Management*, *20*(2), 254-276. https://doi.org/10.14743/apem2025.2.539

[4] Hu, R. & Yang, X. (2025). Enhancing medical big data analytics: A Hadoop and FP-Growth algorithm approach for cloud computing. *Tehnički vjesnik*, *32*(1), 247-254. https://doi.org/10.17559/TV-20240129001302

[5] Bhasker, B., Kaliraj, S., Gobinath, C., & Sivakumar, V. (2025). Optimizing energy task offloading technique using IoMT cloud in healthcare applications. *Journal of Cloud Computing: Advances, Systems and Applications*, *14*(1). https://doi.org/10.1186/s13677-025-00733-0

[6] Puttaswamy, N. G. & Murthy, A. N. (2025). Energy Optimization in Smart Networks using Machine Learning-Driven Fog Computing to Reduce Unnecessary Cloud Data Transmission. Engineering *Technology & Applied Science Research*, *15*(3), 24070-24076. https://doi.org/10.48084/etasr.10236

[7] Amahrouch, A., Saadi, Y., & Kafhali, S. E. (2025). Optimizing Energy Efficiency in Cloud Data Centers: A Reinforcement Learning-Based Virtual Machine Placement Strategy. *Network*, *5*(2), 17. https://doi.org/10.3390/network5020017

[8] Amahrouch, A., Saadi, Y., & Kafhali, S. E. (2025). Optimizing Energy Efficiency in Cloud Data Centers: A Reinforcement Learning-Based Virtual Machine Placement Strategy. *Network*, *5*(2), 17. https://doi.org/10.3390/network5020017

[9] Rasoulpour Shabestari, E. & Shameli-Sendi, A. (2025). An Intelligent VM Placement Method for Minimizing Energy Cost and Carbon Emission in Distributed Cloud Data Centers. *Journal of Grid Computing*, *23*, 12. https://doi.org/10.1007/s10723-025-09798-2

[10] Arega, A. & Sharma, D. P. (2025). Evaluating Energy-efficiency and Performance of Cloud-based Healthcare Systems Using Power-aware Algorithms: An Experimental Simulation Approach for Public Hospitals. *I. J. Information Technology and Computer Science*, *3*, 72-96. https://doi.org/10.5815/ijitcs.2025.03.06

[11] Kotteswari, K., Dhanaraj, R. K., Balusamy, B., Nayyar, A., & Sharma, A. K. (2025). EELB: An energy-efficient load balancing model for cloud environment using Markov decision process. *Computing*, *107*, 81. https://doi.org/10.1007/s00607-025-01439-6

[12] N, N., P, P., & R, A. (2025). Quantum-Enhanced Red Deer Optimization for Optimizing Task Scheduling and Energy Efficiency in Cloud-Based Healthcare Systems. *TPM - Testing, Psychometrics, Methodology in Applied Psychology*, *32*(S3), 2124-2136. https://doi.org/10.1007/s00542-025-05845-4

[13] AlZailaa, A., Ran, H., Radwan, A., & Aguiar, R. L. (2024). Service-aware hierarchical fog-cloud resource mapping for e-health with enhanced-kernel SVM. *Journal of Sensor and Actuator Networks*, *13*(1), 10. https://doi.org/10.3390/jsan13010010

[14] Ganapriya, K., Poobalan, A., Gopinath, S., & Vedha Vinodha, D. (2024). An enhanced trust scheduling algorithm for medical applications in a heterogeneous cloud computing environment. *Tehnički vjesnik*, *31*(3), 945-950. https://doi.org/10.17559/TV-20230913000935

[15] Saad, M., Enam, R. N., & Qureshi, R. (2024). Optimizing multi-objective task scheduling in fog computing with GA-PSO algorithm for big data application. *Frontiers in Big Data*. https://doi.org/10.3389/fdata.2024.1358486

[16] Pan, S., Huang, C., Fan, J., Shi, Z., Tong, J., & Wang, H. (2024). Optimizing Internet of Things fog computing: Through Lyapunov-based long short-term memory particle swarm optimization algorithm for energy consumption optimization. *Sensors*, *24*(4), 1165. https://doi.org/10.3390/s24041165

[17] Thangaraj, V. & Sree, T. R. (2024). MSCO: Mobility-aware secure computation offloading in blockchain-enabled fog computing environments. *Mobile Networks and Applications*. https://doi.org/10.1186/s13677-024-00599-8

[18] Alsamarai, N. A. & Uan, O. N. (2024). Improved performance and cost algorithm for scheduling IoT tasks in fog-cloud environment using Gray Wolf Optimization algorithm. *Applied Sciences*, *14*(4), 1670. https://doi.org/10.3390/app14041670

[19] Abdulazeez, D. H. & Askar, S. (2024). A novel offloading mechanism leveraging fuzzy logic and deep reinforcement learning to improve IoT application performance in a three-layer architecture within the fog-cloud environment. *IEEE Access*. https://doi.org/10.1109/access.2024.3376670

[20] Jagadeeshwar, M., Shobha, V., & Prasad, M. V. S. (2024). Optimized deep neural network based load balancing in fog computing with robust dynamic scheduling algorithm. *Journal of Electrical Systems*. https://doi.org/10.52783/jes.2997

[21] S, M. H. & Gupta, P. (2024). Federated learning inspired Antlion based orchestration for edge computing environment. *PLoS ONE*. https://doi.org/10.1371/journal.pone.0304067

[22] Singhal, S., Sharma, A., Anushree, N., Verma, P. K., Kumar, M., Verma, S., Kavita, N., Kaur, M., Rodrigues, J. J. P. C., Khurma, R. A., & García-Arenas, M. (2024). Energy efficient load balancing algorithm for cloud computing using Rock Hyrax optimization. *IEEE Access*, *12*, 48737-48749. https://doi.org/10.1109/access.2024.3380159

[23] Muniswamy, S. & Vignesh, R. (2024). Joint optimization of load balancing and resource allocation in cloud environment using optimal container management strategy. *Concurrency and Computation: Practice and Experience*, *36*(12). https://doi.org/10.1002/cpe.8035

[24] Srivastava, A. & Kumar, N. (2024). An efficient firefly and honeybee based load balancing mechanism in cloud infrastructure. *Cluster Computing*, *27*, 2805-2827. https://doi.org/10.1007/s10586-023-04118-3

[25] Qasim, M. & Sajid, M. (2024). An efficient IoT task scheduling algorithm in cloud environment using modified Firefly algorithm. *International Journal of Information Technology*, *17*, 179-188. https://doi.org/10.1007/s41870-024-00372-x

[26] Nagarajan, S. M., Devarajan, G. G., Mohammed, A., Ramana, T. V., & Ghosh, U. (2023). Intelligent task scheduling approach for IoT integrated healthcare cyber physical systems. *IEEE Transactions on Network Science and Engineering*. https://doi.org/10.1109/TNSE.2022.3223844

[27] Ahamed, A. M. U., Daniel, D. J. J. D., Seenivasan, D., Khandhan, C. R., Radhakrishnan, S., Sagar, K. V. D., Bhardwaj, V., & Nishant, N. (2023). Deep learning and optimization-based task scheduling algorithms for fog-cloud computing environment. *Journal of Intelligent & Fuzzy Systems*. https://doi.org/10.3233/jifs-234030

[28] Rajpoot, N. K., Singh, P., & Pant, B. (2023). Implementation of delay-sensitive smart healthcare framework in cloud and fog environment. *Research Square*. https://doi.org/10.21203/rs.3.rs-2975684/v1

[29] Fereira, R., Ranaweera, C., Lee, K., & Schneider, J. G. (2023). Energy efficient node selection in edge-fog-cloud layered IoT architecture. *Sensors*, *23*(13), 6039. https://doi.org/10.3390/s23136039

[30] Aqeel, I., Khormi, I., Bhatia, S., Shuaib, M., Almusharraf, A., Alam, S., & Alkhaldi, N. A. (2023). Load balancing using artificial intelligence for cloud-enabled Internet of Everything in healthcare domain. *Sensors*, *23*(11), 5349. https://doi.org/10.3390/s23115349

[31] Vijarania, M., Gupta, S., Agrawal, A., Adigun, M. O., Ajagbe, S. A., & Awotunde, J. B. (2023). Energy efficient load-balancing mechanism in integrated IoT-FOG-Cloud environment. *Electronics*, *12*(11), 2543. https://doi.org/10.3390/electronics12112543

[32] Udayasankaran, P. & Thangaraj, S. J. J. (2023). Energy efficient resource utilization and load balancing in virtual machines using prediction algorithms. *International Journal of Cognitive Computing in Engineering*, *4*, 127-134. https://doi.org/10.1016/j.ijcce.2023.02.005

[33] Tareen, F. N., Alvi, A. N., Malik, A. A., Javed, M. A., Khan, M. B., Saudagar, A. K. J., Alkhathami, M., & Hasanat, M. H. A. (2023). Efficient load balancing for Blockchain-Based healthcare system in smart cities. *Applied Sciences*, *13*(4), 2411. https://doi.org/10.3390/app13042411

Contact information:

P. Manikanda PRABU
(Corresponding author)
Department of Computer Science and Engineering,
Anjalai Ammal Mahalingam Engineering College,
Kovilvenni, Tamilnadu, India
E-mail: mani_adt06@yahoo.co.in

Dr. V. R. Sarma DHULIPALA
Department of Physics,
Anna University, BIT Campus,
Tiruchirappalli, Tamilnadu, India
E-mail: dvrsarma@aubit.edu.in