Journal of Theoretical and Applied Information Technology
31st May 2025. Vol.103. No.10

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

# EDGE LEVEL COMPLEX EVENT PROCESSING AND OPTIMISATION BASED FOG LEVEL LOAD BALANCING MECHANISM IN SOFTWARE DEFINED NETWORK

**SARKARSINHA HARSINHA RAJPUT[*1], DR. MANOJ EKNATH PATIL[2]**

[*1]Ph.D. Research Scholar, Department of Computer Engineering, SSBT's College of Engineering & Technology, Jalgaon, Maharashtra, 425001, India.

[2]Associate Professor, Department of Computer Engineering, SSBT's College of Engineering & Technology, Jalgaon, Maharashtra, 425001, India.

Email: [*1]bs.rajput26@gmail.com, [2]mepatil@gmail.com

**ABSTRACT**

This study uses an efficient transmission model based on the Hybrid Meta-heuristic Model to enhance data transfer by reducing time complexity. Initially, data is moved into the Complex Event Processing (CEP), which is positioned between the fog layer and the IoT layer. In edge IoT devices, complex event processing comprises real-time analysis, correlation, and interpretation of continuous data streams generated by sensors and edge devices. It seeks to identify significant trends or intricate occurrences in various data streams in order to facilitate quick decisions or immediate reactions. After CEP, a multi-tier priority queue-based model is used to attain priority-aware task scheduling. After the arrival of all the tasks, each task is sorted into slots based on its category. High-priority tasks are completed first due to their preference over lower-priority slots. A software-defined network's optimal resource utilization and task response time are guaranteed by an effective load-balancing method called Hybrid Pigeon Cat Search Optimization Algorithms (HPC_SOA). Arranging tasks based on their availability, capacity, proximity, and energy efficiency may optimize the fog nodes' resource utilization and energy usage. In the evaluation, the proposed approach has consumed 22051 Kw/h of energy.

**Keywords:** *Complex Event Processing, Priority Queue Approach, Pigeon Optimization, Cat Search Optimization, Software-Defined Network.*

## 1. INTRODUCTION

Large amounts of data are being produced as industrial processes become more digital, and the complexity of the data is also growing. The employees must be backed by technology in order to extract meaningful information from the massive amounts of created data, as the complexity of the data exceeds human comprehension [1]. Automated techniques for analyzing different types of data processing were needed for human interpretation. Due to the necessity of handling a growing volume of data and promptly responding to data triggers, real-time analytics is becoming progressively vital for enterprises and social applications [2]. The real-time schemes should have the following crucial features for application in industrial fields: low latency, high availability and Horizontal scalability [3]. Intelligent data handling and analysis are required to achieve near-real-time manufacturing and logistics process monitoring and control [4].

This led to the development of Complex Event Processing (CEP), which enables near-real-time processing of massive data streams [5, 6]. CEP refers to the tactics, instruments, and processes used to handle events immediately. CEP's primary purpose is to detect complex event patterns in data streams from sensors and other sources [7]. The basic goal of CEP systems is to create rule forms for similar occurrences based on semantic and spatial correlations. CEP engines rely exclusively on these rules, the vast majority of which are defined by subject matter experts [8, 9]. The complexity of defining and deriving rules is influenced by the industrial procedure and scenario. As a result, one of the limitations that complicates CEP integration and implementation is expert-based rule formulation [10].

The network structure has become increasingly important in recent years due to the rapid growth of traffic rules and network quality, which has caused the network supplies to move quickly [11]. Due to their rigidity, traditional network topologies continue to face challenges in adapting to the dynamic nature of contemporary networks and keeping pace with evolving requirements [12, 13]. Developers created Software-Defined Networking (SDN) to meet the requirement for adaptable networks [14]. SDN enables a more adaptive, scalable, and cost-effective network architecture by separating the control and data planes [15]. The control planes' centralized SDN controller is in charge of packet routing [16]. The data plane is a layer of infrastructure that includes networked forwarding devices such as SDN switches [17].

For SDN-related technologies to be applied effectively, the networking elements must include software in physical architecture [18]. It is the perfect environment for load-balancing implementation because the controller gives information about the network resources that may be employed for load optimization [19]. To avoid network overload, load balancing (LB) is a technique that uses many resources to handle a single operation. In general, LB aims to optimize network traffic without reducing response time and throughput. Load balancing techniques are reputably imprecise in modern networks, but in SDN, they are distinguished by their precision and excellent efficiency [20].

### 1.1 Motivation

The rapid expansion of IoT applications and the increasing demand for real-time data processing have underscored the importance of efficient load balancing and CEP in edge and fog computing environments. Recent literature from 2024 highlights several challenges and advancements in this domain, justifying the need for continued research and innovation. Fog computing serves as an intermediary layer between the cloud and IoT devices, aiming to reduce latency and improve response times. However, the limited resources of fog nodes pose significant challenges in meeting the demands of resource-intensive applications. Efficient load balancing is crucial to prevent overloads and ensure optimal resource utilization. The dynamic nature of IoT applications, characterized by fluctuating workloads and diverse device capabilities, necessitates adaptive scheduling mechanisms. Traditional static load balancing approaches are inadequate in such environments, leading to suboptimal performance and increased latency.

The incorporation of AI techniques, such as deep reinforcement learning and hybrid optimization algorithms, has shown promise in enhancing load balancing strategies. These approaches enable systems to learn and adapt to changing conditions, improving overall efficiency and response times. As fog computing environments handle sensitive data, ensuring secure data transmission and processing is paramount. Additionally, energy efficiency remains a critical concern, especially for battery-powered edge devices. Recent studies have proposed models that address both security and energy consumption, highlighting the need for holistic solutions.

Given these challenges, there is a pressing need for advanced load balancing and CEP mechanisms that can operate efficiently in dynamic, resource-constrained, and heterogeneous fog computing environments. The development of hybrid optimization algorithms, such as the proposed Hybrid Pigeon Cat Search Optimization Algorithm (HPC_SOA), aims to address these issues by combining the strengths of multiple AI techniques. Such approaches can lead to improved resource allocation, reduced latency, enhanced security, and better energy efficiency, ultimately supporting the growing demands of IoT applications. By building upon the insights from recent literature, this study seeks to contribute to the advancement of load balancing and CEP strategies in fog computing, ensuring that future systems are more robust, adaptive, and capable of meeting the evolving needs of real-time data processing. The main contributions of the paper are given as follows.

- To introduce an edge-level complex event processing and optimization-based fog level load balancing mechanism in SDN
- To present a Complex Event Processing (CEP) that entails the real-time analysis, correlation and interpretation of continuous streams of data produced by sensors and edge devices.
- To deploy a Multi-tier priority queue-based model for attaining priority-aware task scheduling.
- To implement the Hybrid Pigeon Cat Search Optimization Algorithms (HPC_SOA) method to build an efficient load balancing mechanism.

In real-time IoT systems, not all tasks are equally critical. CEP enables early identification and categorization of events into simple or complex tasks based on contextual parameters like soil moisture or

temperature. To handle dynamic data streams and maintain processing speed, the CEP module was designed with rule-based logic focusing on frequent event patterns relevant to industrial use cases (e.g., agriculture, monitoring). Once tasks are classified, scheduling must ensure that time-sensitive tasks are prioritized without starving less critical ones. A multi-tier non-preemptive queue ensures fairness and deadline sensitivity.

Utilization factor and task characteristics (e.g., arrival rate, frequency) were used to assign tasks to queues. Task aging and queue rotation were used to prevent starvation and ensure throughput. Traditional optimization methods struggle with either convergence speed or solution quality. HPC_SOA combines the exploration ability of Pigeon Optimization with the local refinement of Cat Search to balance tasks across fog nodes efficiently. Management of Factors: Fitness functions were carefully formulated to incorporate energy usage, response time, proximity, and resource availability. The hybrid algorithm dynamically adjusted parameters (like search range and memory pool) to adapt to workload fluctuations. Most prior works only address specific components (e.g., controller load balancing, or event filtering). To validate practicality, experiments were conducted on a real-world dataset rather than synthetic data, offering realistic variance in workloads and resource constraints. Parameters such as CPU usage, migration time, energy, and response time were closely tracked across different optimization techniques (WOA, SMO, GACO) to ensure a fair comparative analysis.

The paper is organized from Section 1 as the introduction of SDN, and the literature review is given in Section 2. In section 3, the proposed methodology is described, and its evaluation results are given in section 4. The conclusion and future scope are provided in the final section 5.

## 2. RELATED WORK

This section provides a detailed overview of techniques and drawbacks in the existing works on designing an edge-level complex event processing.

For Monitoring the Real-Time IoT, Lan et al. [21] suggested a Universal Edge-Based Complex Event Processing Mechanism. To simplify event modelling, give a defined hierarchical complex event model made up of raw, simple, as well as complex events. The paradigm supports complex time as well as space semantics, allowing programmers to construct flexible, complicated events. A CEP system design was established on the network edge, positioned between cloud applications and terminal sensing devices. The CEP rule logic scripts can be linked to the complicated event description in order to quickly identify any potential anomalous events. The CEP was tiny enough for individual usage and operates on a mobile device.

For Intelligent Agriculture, da Costa Bezerra et al. [22] introduced a Handling Complicated Events in Internet of Things Systems Based on Fog. The new sensor nodes in a network are connected to a Fog node based on the type of data as well as Euclidean distance among them, and utilizing geolocation and context-aware algorithms. This can run simulations in several situations with various network designs and densities to assess the concept. There were some issues on a potential security basis.

Xu et al. [23] suggested Achieving Controller Load Balance in Distributed Software-Defined Networks via Dynamic Switch Migration. To achieve load balancing between SDN controllers with minimal migration costs, the balanced controller (BalCon) as well as BalConPlus SDN switch migration strategies should be suggested. BalCon works well in situations when the network does not need switch requests to be processed in a serial fashion. BalConPlus migrates a small number of switches with minimum calculation overhead, significantly lowering the load imbalance between SDN controllers.

A software-defined network controller adaptive load balancing technique was presented by Priyadarsini et al. [24]. In this study, several switches are migrated from source controllers to target controllers using the self-adaptive load balancing (SALB) technique, which dynamically balances load among different controllers. The ability to disperse load under high load conditions using the estimated distance among target controllers and switches. This previous study has significant limitations due to its complexity and cost.

For Several Controllers in Software-Defined Networking, Li et al. [25] introduced a Fuzzy Satisfaction-Based Load Balancing Method. First, the balancing judgment matrix as well as switch selection degree were introduced to select the transfer of domain as well as migrating switches for controller load monitoring. Second, the migration cost and load balancing rate were considered to be the key load balancing factors. Third, the model was

solved in a short amount of time by choosing the immigration domain using the enhanced ant colony algorithm. The main limitation of the work was handling accurate data was impossible. The abstract view of existing works and their disadvantages are presented in Table 1.

*Table 1: Analysis of existing works and its drawbacks*

| Author name and Reference | Approach used | Performance | Disadvantages |
|---|---|---|---|
| Lan et al. [21] | Universal Edge-Based Complex Event Processing Mechanism | Throughput achieved by the approach is 5.88 Mbps | Relatively high cost |
| da Costa Bezerra et al. [22] | Euclidean distance and context-aware technique | Attain 98% accuracy | There were some issues on a potential security basis. |
| Xu et al. [23] | BalCon and BalConPlus migration strategy | Have 60% of CPU load | Dramatically reducing the load imbalance among SDN controllers. |
| Priyadarsini et al. [24] | SALB | Provide 8.2Mbps of throughput | Complexity and cost were major limitations in this existing work. |
| Li et al. [25] | Fuzzy Satisfaction-Based Load Balancing Method | The average response time of controllers by about 0.33 s | Handling accurate data was impossible. |

From the analysis of existing models, these models have demerits, like relatively high cost in implementation. Some models have issues on a potential security basis, and some have dramatically reduced the load imbalance among SDN controllers. Moreover, complexity and cost were major limitations in this existing work. On the other hand, handling accurate data was impossible. Hence, the proposed model has been implement to overcome the existing flaws.

### 2.1 Research Gap and Problem Statement

Despite the significant advancements in edge computing, fog computing, and Software Defined Networking (SDN), several limitations continue to persist in the existing load balancing and event processing techniques. Most existing works focus either on real-time event processing or on load balancing in SDN-based systems, but seldom integrate both into a unified framework. This disjointed approach leads to inefficiencies in task scheduling and decision-making at the fog and edge layers.

Techniques like BalCon, SALB, or fuzzy satisfaction models tend to rely on predefined rules or lack adaptability, making them ineffective in highly dynamic environments with fluctuating workloads and heterogeneous resources.

Although heuristic methods like Ant Colony or Genetic Algorithms are used, they often suffer from convergence issues and lack global optimization capabilities. There is a lack of comprehensive hybrid meta-heuristic models that intelligently combine exploration and exploitation for optimized task distribution.

Existing models do not fully utilize context-based task categorization for priority-aware scheduling. This leads to increased task migration, energy consumption, and delayed response time. Based on the identified gaps, the central research question addressed by this study is "How can an integrated, hybrid optimization-based model combining Complex Event Processing and priority-aware scheduling enhance task allocation and load balancing in a fog-enabled SDN environment to reduce energy consumption, migration time, and response delay?" This work addresses the gap by proposing a novel HPC_SOA integrated with CEP and a Multi-tier Priority Queue Scheduling system. This design ensures efficient task categorization, reduced time complexity, intelligent load balancing, and optimal resource utilization in dynamic IoT environments.

## 3. PROPOSED METHODOLOGY

The rapid development of technology, along with cheaper costs and more connection capacity, have contributed to the IoT's enormous rise in popularity. Infrastructure and data expansion must be matched by a software architecture that facilitates their exploitation. It is challenging to find a software solution that fully utilizes contextual and situational information processing despite numerous proposals focusing on the edge and fog levels. This study uses an efficient transmission model based on the Hybrid Meta-heuristic Model to enhance data transfer while reducing time complexity. The schematic representation of the process flow of the suggested approach is shown in Figure 1.

A publicly accessible dataset was used to acquire the data. Initially, the data is moved into the CEP, which is positioned between the fog layer and the IoT layer. Alert will be displayed after every sample of the dataset is processed. In edge IoT devices, complex event processing entails the real-time analysis, correlation and interpretation of continuous streams of data produced by sensors and edge devices. It seeks to identify significant trends or intricate occurrences in various data streams in order to facilitate quick decisions or immediate reactions. After CEP, a multi-tier priority queue-based model is used to attain priority-aware task scheduling. Upon arrival, a task from an application is sorted into slots based on its category. High-priority tasks are finished first because they are in higher-priority slots than lower-priority slots.

Tasks in lower slots are executed only if upper slots are empty. Task execution follows a non-pre-emptive approach, ensuring that the current task is completed before the next one is selected. The major goal of the load balancing mechanism on a fog server in a software-defined network is to improve the fog computing system's performance, efficiency, and reliability. The system model of the proposed Fog Level Load Balancing Mechanism is illustrated in Figure 2.

This system supports a range of IoT applications that need improved security, high scalability and low latency. A software-defined network's optimal resource utilization and task response time are guaranteed by the HPC_SOA approach. Arranging tasks based on their availability, capacity, proximity, and energy efficiency may optimize the fog nodes' resource utilization and energy usage. A detailed description of the approaches is given in the further sections.

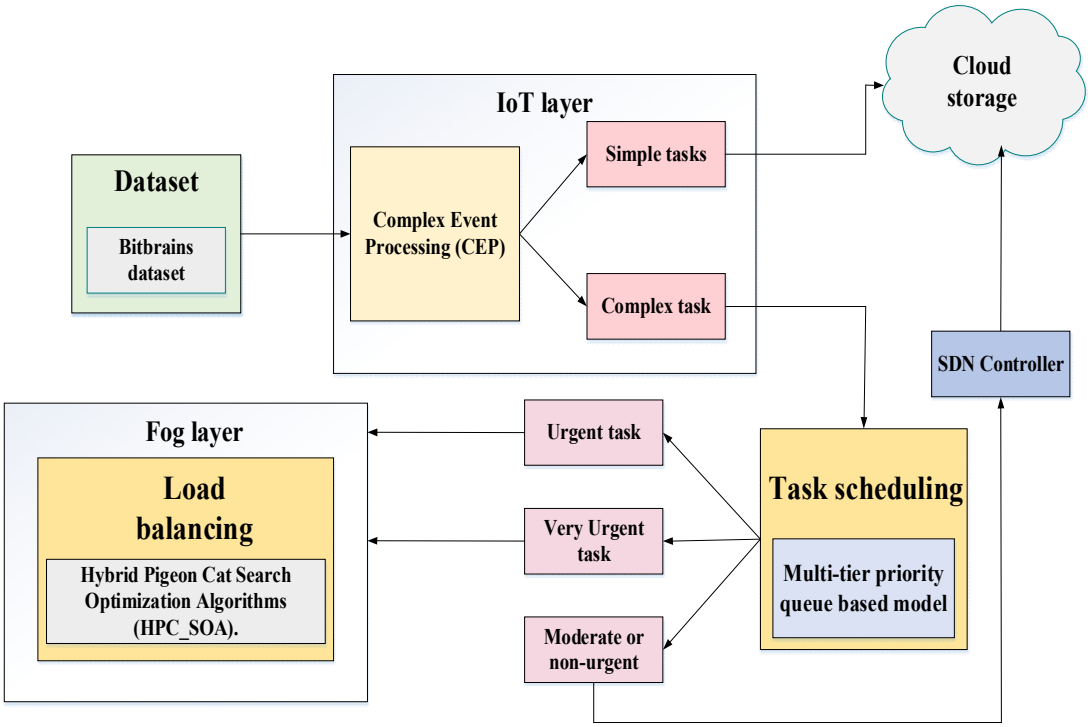


*Figure 1: Schematic Representation Of The Process Flow Of The Proposed Model*

Journal of Theoretical and Applied Information Technology
31st May 2025. Vol.103. No.10

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

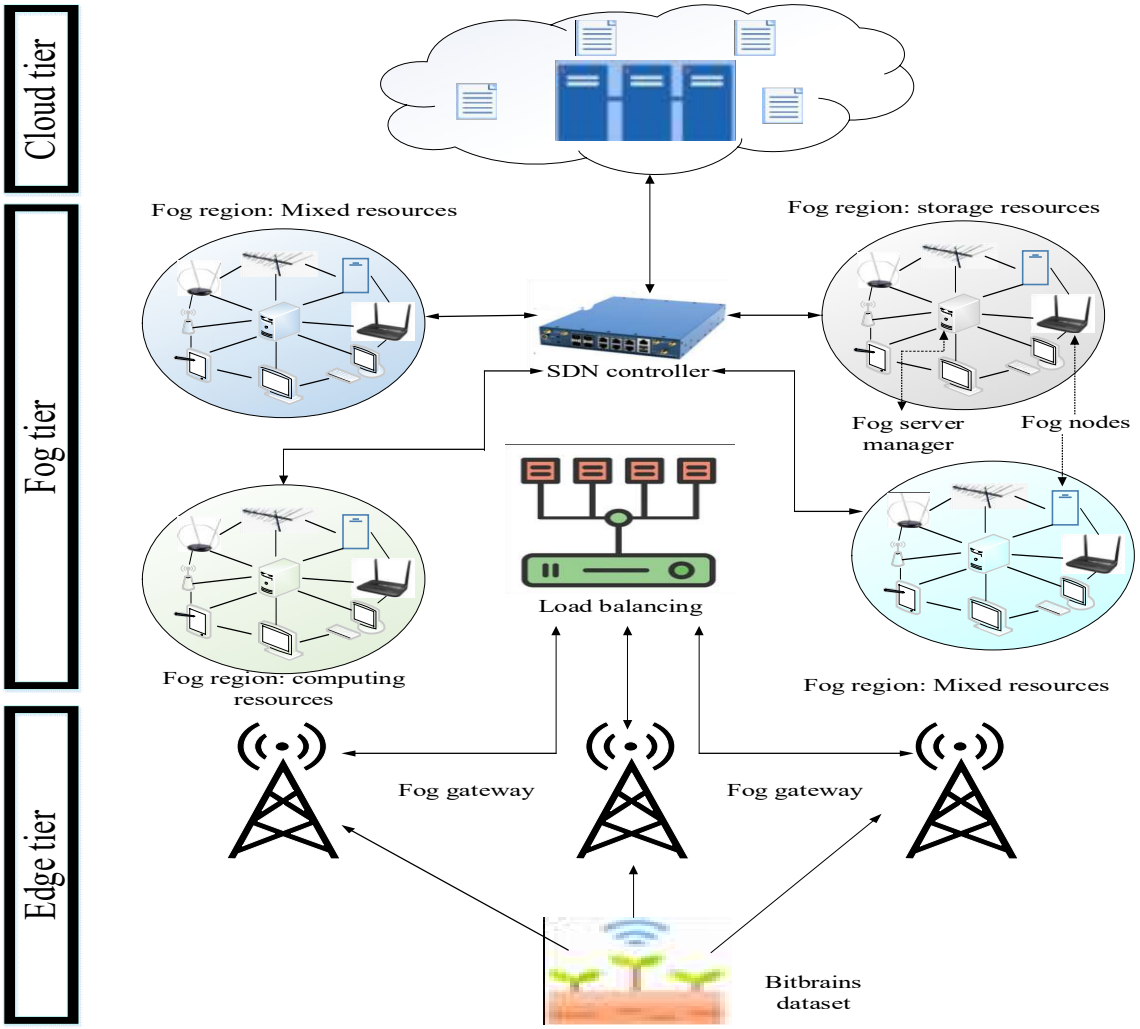


*Figure 2: Proposed System Architecture For Load Balancing In Fog Layer*

### 3.1 Complex Event Processing

The tasks are divided into simple and complex events using the rules of Complex event processing (CEP). Complex event processing [26] is a computing technique that analyses data streams in real-time to identify and understand the events. It is event-driven, meaning it is triggered by the receipt of event data. Consider an agriculture-related event, which is classified as simple or complicated based on complex event processing. The production goal in the agricultural industry is to maximize crop yield at the lowest possible cost. Making decisions in this situation is complex since several variables influence the entire process, with the primary goal of conserving water for irrigation. To meet the requirements of the atmosphere, a growing plant must extract water from the soil.

Evapotranspiration reduces the amount of water stored in the soil; precipitation or irrigation takes its place. Therefore, it is important to characterize the soil features responsible for water retention. Moisture and matric potential are intimately tied to the soil's potential, which increases with humidity. A crop's ability to store enough water during its development is aided by irrigation. When managing irrigation, its matric potential is utilized to choose the kind, quantity, and timing of irrigation. The soil matric potential $\psi$ containing the highest concentration of crop roots. Another method for figuring out when the optimal moisture for irrigation, or critical moisture $\theta$ is where irrigation should take place. This metric represents the soil water content at which crop yields begin to decline with the potential for a decrease in evapotranspiration. $\theta$

required for the soil tier which is the most shallow feature. Based on the given concepts, the rules for the process are formed. Normally, there are two types of rules in the CEP such as monitoring rules as well as critical state rules.

**Monitoring rules:** The foundation of the first set of events in the monitoring rule is provided by the primitives of filtration, aggregation, as well as projection. Using a portion of its resources, the projection generates complex events that are handled by the fog tier. Characteristics that are produced by filtering and aggregating. The following incidents made up the first set's rule:

- A low matric potential alert is sent by EV1 (Simple)
- A high matric potential alert is sent by EV2 (Simple)
- A daily matric potential sending alert plus sensor ID is sent by EV3 (Simple).

The second level of events is made up of both simple and complex events that are built on the Fog tier's aggregation, enrichment, and composition primitives. The gathering of information needed for computation is referred to as enrichment. Composition refers to the computations made using the data that were examined. The following incidents make up this rule:

- EV5 (complex): daily irrigation water requirement (IWN)
- EV6 (complex): daily irrigation frequency verification
- EV4 (complex): matric potential variation.

**Critical state rules:** The critical state rule consists of events based on the primitives of enrichment, composition, negation, and sequencing performed at the Fog tier. Implementing a sliding time window with an event identification and a timestamp integrated will provide the input control required to use the sequence primitive. The following are the events that make up this rule:

- EV7 (Complex): maximum water deficit
- EV8 (Complex): essential soil moisture (perfect for irrigation).

According to the given events, the proposed rules of the CEP technique for separating simple and complex events are stated in the pseudocode provided in Table 2.

*Table 2: Pseudocode Of CEP Approach*

```
Start
Initialize the task counts and task
Apply the CEP rules
For task i
        If soil humidity < 50
                Update as a complex task
        Else if wind speed (Km/h) < 9
                Update as a complex task
        Else if Irrigation field < 1.0
                Update as a complex task
        Else if Wind direction (Deg) < 93
                Update as a complex task
        Else if Air temper ature (C) < 50
                Update as a complex task
        Else if Pressure< 101.3
                Update as a complex task
        Else
                Update as a simple task
        End If
Stop
```

The CEP of tasks are done in the edge layers, and the complex tasks are sent to the fog layer and then scheduled.

### 3.2 Multi-Tier Priority Queue-Based Model for Priority Aware Task Scheduling

After separating the complex task using CEP, the priority task scheduling [27] will take place to schedule the task by its importance. Initially, consider the scheduled system $S = S_1, S_2, \ldots, S_m$ in which the Poisson distribution method is used to distribute inter-arrival jobs between two successive tasks. These distribution is given as $F(Y) = \Pr(Y = y) = e^{-\lambda}\lambda^{y} / y!$ here $e$ represents the Euler's No, $\lambda > 0$ and $Y$ represents mean arrival rate. According to the suggested sensor network design, each gateway device assigns tasks to four categories: very urgent, urgent, moderate, and non-urgent. Very Urgent is given high priority since it aims to assist jobs with short deadlines that must be finished fast. Soil humidity, air humidity, and air

temperature are considered to be the most urgent duties from complicated events, with the remaining tasks being urgent. Furthermore, basic events are categorized as moderate or non-urgent occurrences. To avoid long offloading and downloading periods in an SDN cloud, workloads are sent to neighbouring edge devices or fog layers. The architecture of the Multi-tier priority queue-based model for Priority-aware task scheduling is provided in Figure 3.

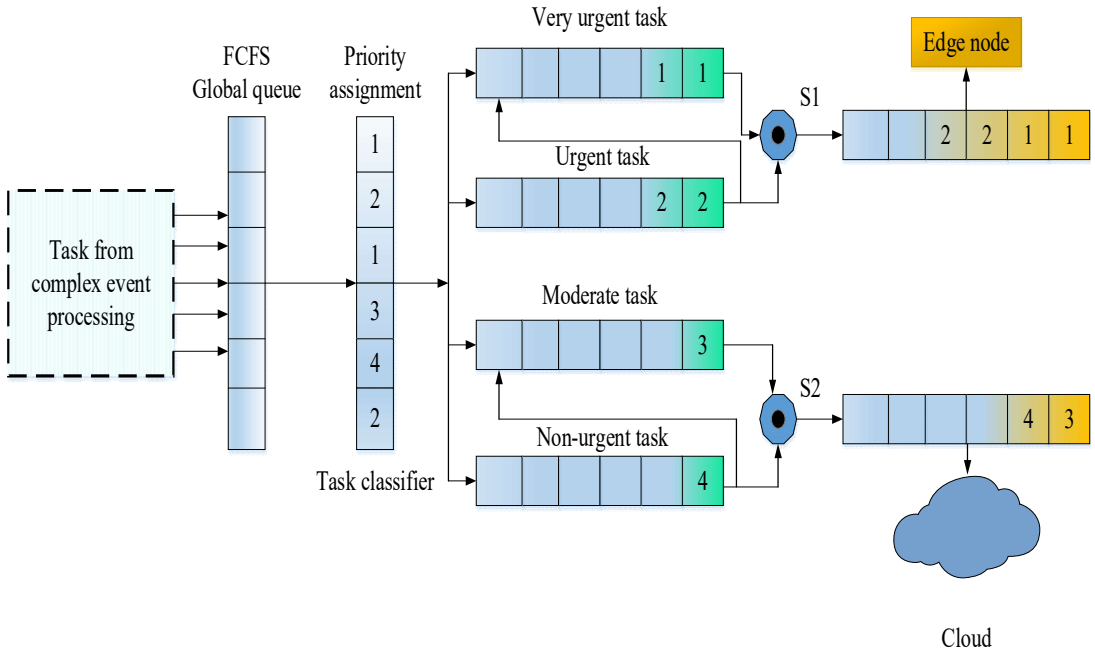


*Figure 3: Architecture Of Multi-Tier Priority Queue-Based Model For Priority-Aware Task Scheduling*

Sometimes, a more dependable computing device is designated especially for these tasks. An urgent task's structure is essentially equal to that of a very urgent task; however, an urgent task necessitates a little extra waiting time. Moderate tasks have a moderate priority level and a flexible time limit. These tasks negotiate total latency, which means that more time may be provided to complete the task processing in order to meet the deadline, and these jobs are processed by the cloud server. The lowest priority is given to the non-urgent tasks, which are expected to allow time-consuming jobs with no deadline. These tasks require more processing power without a priority on reducing latency. These kinds of tasks are usually offloaded to a centralized cloud server. In order to facilitate additional decision-making and minimize scheduling delays, the gateway devices create non-preemptive four-level feedback queues. These queues are said to be Very Urgent (QVU), Urgent (QU), Moderate (QM), as well as Non-Urgent (QNU). Work is done on the jobs and queue using a priority-aware task scheduling methodology. The task will be removed from any queue if the waiting time reaches a certain level in order to prevent starvation. When the extremely urgent task queue is empty, the task from the urgent task will be moved to the very urgent task queue.

This scheduling algorithm's main job is to use the Utilization Factor $Uf_j$ to classify the incoming tasks generated by sensor devices and then assign those jobs to the appropriate queues according to priority. Each task $T_j$ in $T$ is assigned a task identifier from a collection of emergency information, such as the task's execution deadline $T_j^{ed}$ and frequency of arrival $T_j^f$. Utilization factors $Uf_j$ govern the priority of each incoming job and are expressed as $Uf_j = T_j^{ed} / T_j^f$. Then, the tasks are classified as Very Urgent $T_j^{vu}$, Urgent $T_j^u$, Moderate $T_j^m$ or Non-Urgent $T_j^{nu}$, depending on

this usage factor. Sensor device-generated tasks will be prioritized in the global queue according to the Gateway's First Come First Served (FCFS) order (i.e., Raspberry Pi).

The work will be given an additional priority depending on its utilization factor, and subsequent tasks will be scheduled into the appropriate queues in accordance with their Priority Aware Scheduling Policy. To prevent starvation, a task will be removed from a queue if its waiting time in that queue exceeds a certain threshold. Two pairs of queues were added to the queue model to provide some variety. One set of queues handled urgent work, which was subsequently offloaded to the fog server, and the other pair handled non-urgent activities, which were handled directly by the cloud's offload server. Priority assignment and queue allocation will occur within the gate device. The task priority is defined by the given definition.

**Definition:** From the sensor device, the task $T_j$ would arise and be defined as a very urgent task if $Uf_j < \frac{1}{4}$ or moderate if $\frac{1}{3} < Uf_j < \frac{1}{2}$ or urgent if $\frac{1}{4} < Uf_j < \frac{1}{3}$ or non-urgent if $Uf_j > \frac{1}{2}$.

**Explanation:** Assume six sets of tasks that are said to be $T_j = T_1, T_2, T_3, T_4, T_5, T_6$ which has set as $T_1 =< 4,4 >$, $T_2 =< 4,6 >$, $T_3 =<3,8>$, $T_4 =< 2,9 >$, $T_5 =<1,2>$, $T_6 =<4,5>$. According to the utilization factor norms, Task $T_j$ utilization is estimated using the expressions as $Uf = 1$. Similarly, $Uf = 0.6$, $Uf = 0.3$, $Uf = 0.2$, $Uf = 0.5$ and $Uf = 0.8$. Task $T_4$ is classified as a very urgent task since task priority is determined by utilization level. According to the definition, tasks $T_1$, $T_2$ and $T_6$ are categorized as non-urgent, $T_3$ as urgent, and $T_3$ as moderate.

### 3.2.1 Scheduling task

In order to create a multilevel feedback Queue model, a Multi-tier priority queue-based model takes into account four queues. The most urgent activities are sent to the Very High Priority queue, which is represented by $qu1$. Similar to $qu1$, urgent jobs are routed to $qu2$ which is a lower-priority queue. Non-urgent tasks were routed to the Low Priority Queue $qu3$, while moderate tasks were directed to the queue $qu4$ with moderate. $deq()$ is also used to establish a feasible schedule order for data reception based on priority.

**On scheduler 1:**

**Rule 1:** When $qu2^u \neq \varphi$ and $Wr_j^{Qt}, ^u \geq qu'^u, \forall_j \in T_j{}^u$ task $T_j$ need higher priority than given by pushing in $T_j^{vu}$. Replace the task to a very urgent queue from an urgent queue based on FCFS, which is expressed as $deq(T_j^u)qu^{vu}$.

**Rule 2:** If $qu1^{vu} = \varphi$ and $qu^{vu} \neq \varphi$, $deq(T_j^u)$ from urgent queue to load directly on fog server based on FCFS.

**Rule 3:** If $qu1^{vu} \neq \varphi$ and $qu^u = \varphi$, according to the FCFS approach, the task is removed from a very urgent task using $deq(T_j^{vu})qu^u$ to load at fog server by the gateway.

**At scheduler 2:**

**Rule 1:** If $qu4^{nu} \neq \varphi$ and $Wr_j^{Qt}, ^{nu} \geq qu'^u, \forall_j \in T_j{}^{nu}$ task $T_j$ need higher priority, that is given by pushing in $T_j^m$. Replace the task to moderate queue from non-urgent queue based on FCFS, which is expressed as $deq(T_j^{nu})qu^m$. Here, $Wr_j^{Qt}$ denotes the time quantum

**Rule 2:** If $qu3^m = \varphi$ and $qu1^{nu} \neq \varphi$, after that, offload $T_j$ to the cloud using Priority $T_j^{nu}$ to de-queue it in FCFS sequence.

**Rule 3:** If $qu3^m \neq \varphi$ and $qu1^{nu} = \varphi$, offload $T_j$ to the cloud using Priority $T_j^m$ to de-queue it in FCFS sequence.

The algorithm using a Multi-tier priority queue-based model for Priority-aware Task Scheduling is presented in Table 3.

*Table 3: Algorithm for Priority-Aware Task Scheduling*

**Input:** Set of tasks $T_j$, attributes of task
**Output:** Respective Task schedule
for $T_j \leftarrow 1$ to $T$ do
Compute utilization factor $Uf_j = T_j^{ed} / T_j^f$
Provide priority for every task $T_j$ according to $Uf$
Add Every task $T_j$ to the priority queue $qu$
Allocate task for $qu^{vu}$ $qu^{u}$ $qu^{m}$ and $qu^{nu}$ by $Uf_j \le \frac{1}{4}, \frac{1}{4} < Uf_j < \frac{1}{3}, \frac{1}{3} < Uf_j < \frac{1}{2}$ and $Uf_j > \frac{1}{2}$
**end for**
**for** $T_j \leftarrow 1$ to $T$ **do**
initialize time quantum $Wr_j^{Qt} = 0$
prioritize and schedule the task $T_j$
Later certain time quantum $deq(T_j)$
Update $Wr_j^{Qt} = Wr_j^{Qt} + qu'$
**end for**

According to the algorithm, tasks are scheduled between the fog layer and edge layer, where the urgent and very urgent tasks are sent to the fog layer, and moderate or non-urgent tasks are transferred to the cloud. Then, the load-balancing approach in the task takes place using the proposed hybrid Meta heuristic algorithm.

### 3.3 Load Balancing Using HPC_SOA Approach

The complex tasks from the IoT layer are balanced to facilitate the equal distribution of workload on resources. The load balancing approach is done by Hybrid Pigeon Cat Search Optimization Algorithms (HPC_SOA). Here, the exploration stage of both pigeon [28] and cat optimization [29] are combined together, and the fitness function for load balancing is defined. Using the function loads i.e., the tasks are balanced in the fog layer. Initially, the population of loads are defined using the equation 1.

$$L_{j,\dim}^{ig} = Bd_{\min} + rd\left(Bd_{\max} - Bd_{\min}\right) \quad (1)$$

Here, the targets are indicated as $L_{j,\dim}^{ig}$, $j$ represents the initial generation, and the current dimension is given by $\dim$. The maximum bound and minimum bound values are denoted by $Bd_{\max}$ and $Bd_{\min}$ respectively. The load balancing is done using the seeking mode strategy of cat optimization. This mode, which resembles cat sleeping behaviour, is made up of four important factors: counts of dimension to change (CDC), seeking range of the selected dimension (SRD), seeking memory pool (SMP), as well as self-position consideration (SPC). SMP determines the amount of cat copies (candidate places) in the search mode. For example, if SMP is set to 5, each cat in the searching mode will generate five random copies, one of which will be selected and moved to the next position. These random copies are produced in a manner that is dependent upon SPC and CDC. The CDC parameter indicates how many dimensions need to be changed and is between 0 and 1. Finally, SPC is a Boolean variable that controls if the cat's current location is utilized as one of the SMP copies. For example, if SPC is set to true as well as SMP is set to 5, only four random copies will be made, with the current location displayed. Assume to be the $5^{th}$ applicant for the job. The following are the steps in this mode:

1. Copy a cat's current location $i$ times where $i = SMP$. If SPC is true, then $i = SMP - 1$ and remain a candidate in current position.
2. The CDC recommends changing a few random dimensions for every copy. Next, arbitrary SRD values should be added or subtracted from the current positions by the equation (2).

$$Y_{i,\dim} = \left(1 \pm rd * SRD\right) * Y_{I,\dim} \quad (2)$$

Here, the position of the cat is said to be $Y_{i,\dim}$ where $i$ gives the copies of the cat.

3. Estimate the cost value of the candidate position.
4. Determine each candidate point's selection probability using the roulette wheel approach by using equation (3). As a result, Candidate points with higher fitness costs stand a higher probability of being chosen. On the other hand, the selection probability of every candidate point would be 1 if all fitness costs were equal.

$$\Pr_j = \frac{\left|F_j - F_c\right|}{F_{\max} - F_{\min}} \quad (3)$$

Here, $\Pr$ represented the selection probability and will be the fitness when the objective

is minimum the value of $F_c$ be $F_{max}$ while maximum it will be $F_{min}$ . This fitness function is calculated by the parameters of nodes.

In the fog layer SDN-based load balancing, fitness functions are utilized to compute performance. The major variables used to calculate fitness based on the HPC_SOA technique are response time, energy efficiency, resource usage, and throughput. The following equations are used as a proposed fitness function.

$$Responsetime = end\_of\_processing - start\_of\_processing \quad (4)$$

$$Energy\_Eff = \frac{total\_workload}{Energy\_consumed} \quad (5)$$

$$Throughput = \frac{number\_of\_request\_sent}{time\_period} \quad (6)$$

$$resourse\_utilization = \frac{utilized\_resources}{total\_available\_resources} \quad (7)$$

Using the fitness function, the provisioning and de-provisioning tasks are done optimally, where the load balance is provided efficiently in the Fog layer. The results from the fitness values are used to decide whether the load is overloaded or underloaded. When the fitness value is above 75%, it will be said to be overloaded; when the fitness value is under 25%, it will said to be underloaded. The evaluation results of the suggested approach are provided in the further section.

## 4. RESULT AND DISCUSSION

The proposed approach of the Fog Level Load Balancing process is implemented using Java with the iFogSim tool. The system configuration used for the implementation is shown in Table 4.

*Table 4: System configuration*

| Components | Details |
|---|---|
| Processor | Intel(R) Core(TM) i3-3245 CPU @ 3.40GHz 3.40 GHz |
| Installed RAM | 8.00 GB (7.83 GB usable) |
| System type | 64-bit operating system, x64-based processor |
| Pen and touch | No pen or touch input is available for this display |

### 4.1 Bitbrains Dataset Description

The dataset includes the performance parameters of 1,750 virtual machines (VMs) from Bitbrains' scattered datacenter, which provides business computing and managed hosting of Numerous large banks (ING), credit card companies (ICS), insurers (AEGON), etc among the customers. Apps utilized in the solvency domain are hosted by Bitbrains providers of apps, including Algorithmic and Towers Watson. Financial reporting is the usual purpose for these apps, and it happens mostly at the end of the fiscal quarter. Every file contains a virtual machine's performance metrics. The files are arranged according to traces: Rnd and fastStorage. 1,250 VMs in the first trace and the fast storage are linked to storage devices via a fast storage area network (SAN).

The 500 VMs in the second trace, Rnd, are either connected to faster SAN devices or much slower Network Attached Storage (NAS) devices. Considering that storage linked to fast storage devices performs better than the Rnd trace. The fastStorage trace includes a larger percentage of application servers as well as compute nodes. On the other hand, the larger percentage of management machines for the Rnd trace need less-performing storage that is accessed less frequently. The files in the Rnd directory are arranged into three subdirectories according to the month in which the metrics are captured [30].

### 4.2 Performance Analysis

The performance of the suggested approach is measured by evaluating several metrics like Scheduling time, energy consumption, number of tasks migrated, average migration time, total migration time, CPU utilization, response time and simulation time. This is proved by comparing the results with other current methods that are implemented.

#### 4.2.1 CPU utilization

CPU utilization is said to be the amount of work a CPU does to perform the load-balancing tasks given. The estimation of CPU utilization is made using the equation below.

$$CPUutilization = TotalCPUcapacity - CPUconsumed \quad (8)$$

#### 4.2.2 Response time

Response time is said to be the time taken to respond to the server and client, which is estimated by the following equation.

$$Responsetime = end\_of\_processing - start\_of\_processing \quad (9)$$

#### 4.2.3 Energy consumption

Energy consumption is the entire amount of energy required for a load-balancing operation and is expressed in kilowatt-hours (kWh). The equation used for the estimation of energy consumption is given as follows.

$$Energyconsumption = InitialEnergy - Energyconsumed \quad (10)$$

#### 4.2.4 Scheduling time

Scheduling time is defined as the time taken by the proposed technique to schedule the task in the network.

#### 4.2.5 Number of tasks migrated

The number of tasks migrated refers to the number of tasks that are migrated from the edge layer to the fog layer during the CEP and scheduling processes.

#### 4.2.6 Average of migration time

The average migration time by the migration nodes for migration from one layer to another layer.

#### 4.2.7 Total migration time

The total taken for migration time by the migration nodes for migration from one layer to another layer.

### 4.3 Performance Evaluation

The scheduling time of the suggested model is compared with other models, as shown in Figure 4.

The proposed approach possesses less scheduling time than other approaches, such as the Whale optimization algorithm (WOA), Spider monkey optimization (SMO) and Genetic ant colony optimization (GACO). The scheduling time taken by the proposed model is 8.123 sec, while the other approaches have taken 13.568 sec, 18.254 sec and 20.356 sec. Among the existing works, the given approach provides minimum time for scheduling due to the categorization of tasks by using the CEP module and then scheduling by a Multi-tier priority queue-based model. The comparative analysis of average response time by the suggested model is illustrate in Figure 5.

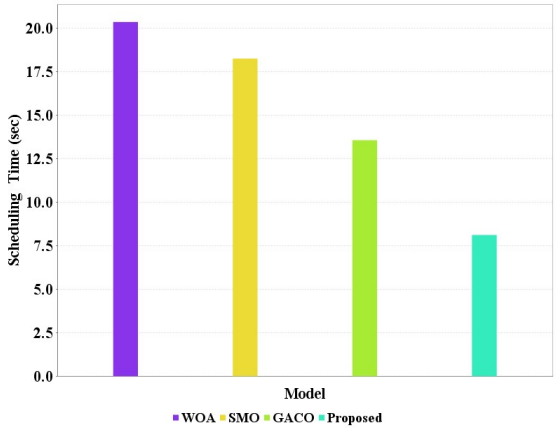


*Figure 4: Scheduling time analysis of the proposed mechanism*

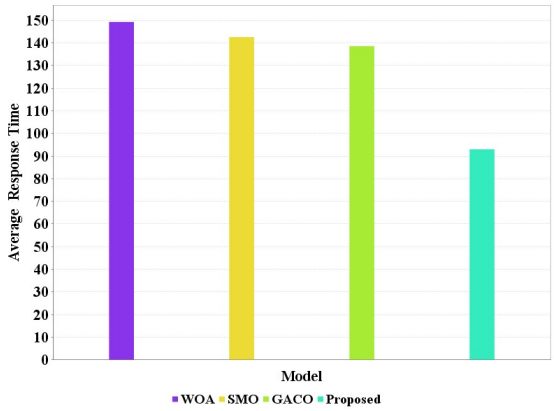


*Figure 5: Analysis of average response time by proposed vs. existing*

The average response time for communication of tasks when using the suggested approach is less as compared to existing approaches. The existing approach considers for comparing the time taken are WOA, SMO and GACO. The average response time taken by the suggested approach is 81sec while other models were taken about 131.58 sec, 142.556 sec, and 149.25 sec, respectively. The response time of the suggested approach is less than that of other existing approach due to its task-scheduling mechanism. When compared to other methods, the fog layer stores the most significant tasks that are appropriately important and take less time to access. The comparative analysis of the migration of tasks by the suggested model and the existing model is presented in Figure 6.

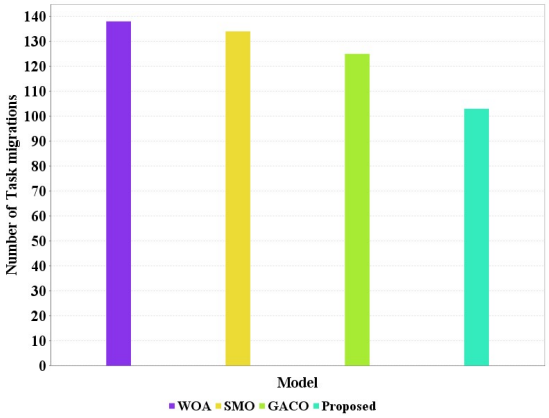


*Figure 6: Comparative analysis in migration of task by the proposed model vs. existing models*

Using the suggested approach, most of the tasks are completed within their scheduled time without migration. The proposed model has a migrated task of about 103, while other models have 125 tasks, 134 tasks and 138 tasks, respectively. The proposed model has less migration tasks due to its scheduling approach; the task gets completed one by one in its scheduled timing. The time taken for task migration using suggested method is compared with the current one and illustrated in Figure 7.

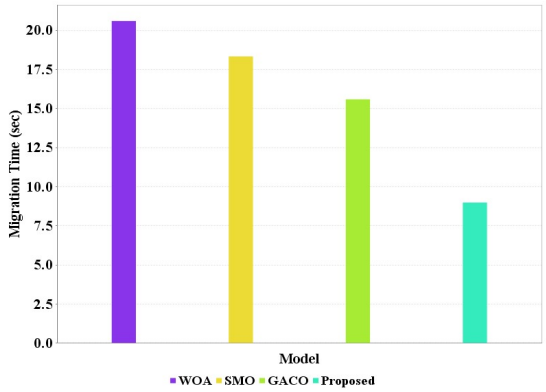


*Figure 7: Analysis of migration time taken by the proposed model*

The amount of time required for a task to be completed during a migration is referred to as the migration time. Using the proposed approach, the task migration time is less due to an efficient task scheduler. The migration time utilizing the proposed technique is 8.985 sec, whereas the other models take 15.584 sec, 18.326 sec, and 20.589 sec. The categorization of complex and simple tasks makes the scheduling task easier, where the tasks get completed earlier at the scheduled time. The average migration time taken to show the efficiency of the proposed approach is presented in Figure 8.

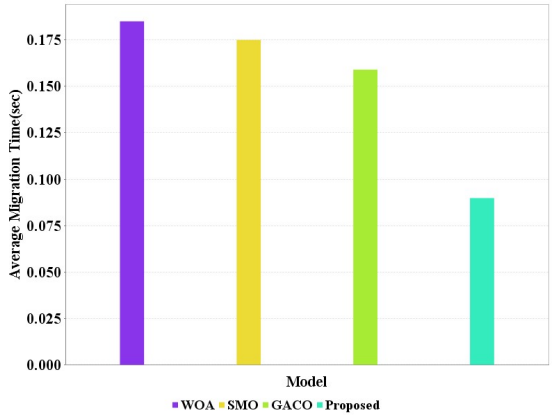


*Figure 8: Analysis of average migration time taken by the proposed model*

The average migration time is defined as the amount of time spent on a task without completion during the migrating process. Using the proposed approach, the task migration time is less due to efficient task scheduling. The average migration time utilizing the proposed technique is 0.08985 sec, whereas the other models take 0.159 sec, 0.175 sec, and 0.185 sec, respectively. The categorization of complex and simple tasks makes scheduling tasks easier, allowing tasks to be completed earlier at the scheduled time. The evaluation of energy consumption is taken to show the efficiency of the suggested method, is illustrate in Figure 9.

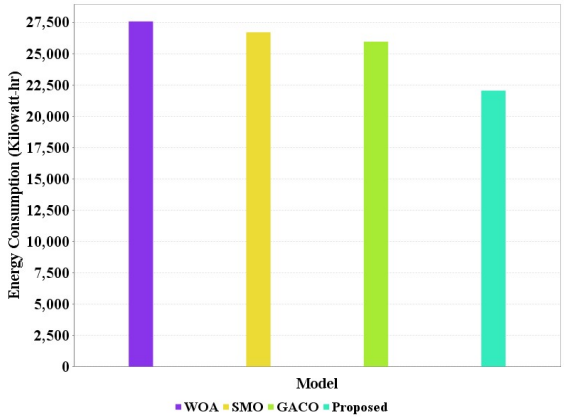


*Figure 9: Comparative analysis of energy consumption*

The energy consumed by the proposed strategy in load balancing is less than that of other models. The energy consumed by the suggested approach is 22051.788 Kw/h, while the other models consume 25950.59 Kw/h, 26698.26 Kw/h and 27565.32 Kw/h, respectively. The consumption of

energy is lower due to the categorization of complex tasks and schedules by using CEP and a multi-tier priority queue-based model. The scheduled tasks are then balanced so that the consumption of energy is less by the proposed approach. The utilization of the CPU is also analyzed and depicted in Figure 10.

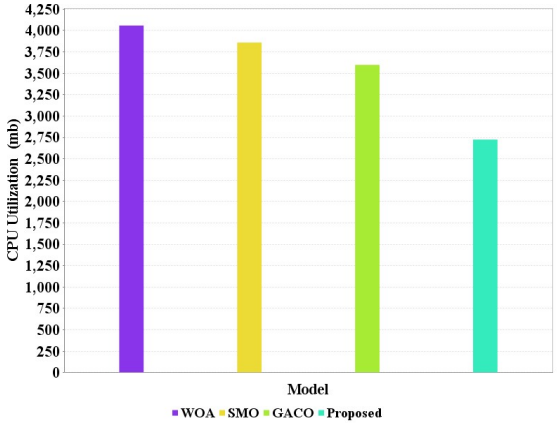


*Figure 10: CPU utilization by the proposed model vs. existing models*

The CPU utilization of the proposed model is less than that of existing models such as WOA, SMO, and GACO. The proposed model utilizes a CPU of about 2723.06MB, while the existing models use 3598.249 MB, 3859.524 MB and 4058.362 MB, respectively. The load balancing methodology by HPC_SOA provides efficient balancing in the fog layer while CPU usage is less than that of other models.

### 4.3.1 Ablation study

The performance of the model with and without the CEP approach is analyzed, and the ablation study of the suggested methodology is presented in Table 5.

In the evaluation of the proposed approach using CEP and without CEP, there will be a decrement in the performance of the model. The Scheduling time of without CEP approach is higher than with CEP approach due to non-specification and disorder of tasks. In the evaluation of CPU utilization without CEP process, the utilization is higher due to the processing of every task in the fog layer. For these reasons, the model consumes more energy than CEP-based methodologies. The migration time and number of migrations increase in the absence of a CEP process due to inappropriate load balancing caused by task non-specification and disorder in scheduling.

*Table 5: Ablation study of the proposed model*

| Task | Scheduling time (sec) | CPU utilization (MB) | Average migration time (sec) | migration time (sec) | Response time (sec) | Number of tasks migrated | Energy consumption $kw/h$ |
|---|---|---|---|---|---|---|---|
| With CEP | 8.123 | 2723.06 | 81.0 | 8.985 | 8.123 | 103 | 22051 |
| Without CEP | 8.828 | 2898.06 | 0.09560 | 9.524 | 90 | 110 | 25846.788 |

### 4.4 Discussion

Despite the many suggestions that concentrate on the edge and fog levels of the IoT, it is a difficult challenge to develop a software solution that fully uses contextual and situational information processing at every level as well as two-way communications between nearby ones. The data is initially transferred into the CEP. In edge IoT devices, CEP comprises the real-time correlation, analysis, and interpretation of continuous data streams generated by edge devices and sensors. In order to support prompt judgments or prompt replies, it looks for noteworthy trends or complex occurrences in diverse data sources. A multi-tier priority queue-based approach is employed to achieve priority-aware task scheduling after CEP. Upon every task's arrival, a task from an application is sorted into slots based on its type. Since high-priority jobs are preferred over lower-priority slots, they are completed first. HPC_SOA is an efficient load-balancing technique that ensures the best possible resource usage and task response time for a software-defined network. It might maximize the fog nodes' resource and energy consumption by allocating jobs according to their capacity, availability, proximity, and energy efficiency. In the evaluation, the proposed model has a scheduling time of 8.123 sec, a response time of 81 sec, and an average time of migration of 0.08985 sec. Also, the total time taken by the proposed model of task migration is evaluated, and the results are 8.985 sec, and the tasks that are migrated by the proposed model are 103. The energy consumption of the proposed model is estimated, which yields 22051 $kw/h$, and the CPU utilization is checked and

results as 2723.06 MB. The scores attained by the proposed approach show that the model would perform much better than existing models. The proposed hybrid optimization-based framework for load balancing and event processing has several important implications for real-world industrial applications. In sectors such as smart manufacturing, precision agriculture, healthcare monitoring, and autonomous logistics, the timely processing of sensor-generated data and intelligent task scheduling are critical. By integrating CEP at the edge level, the system enables near real-time detection and reaction to key events (e.g., equipment failure, soil moisture depletion, abnormal patient vitals), reducing dependency on centralized cloud processing and latency. The multi-tier priority-aware scheduling ensures that urgent and critical tasks are handled with minimal delay, which is vital in time-sensitive environments. Furthermore, the HPC_SOA enhances fog-level resource management, making it feasible to deploy scalable and energy-efficient IoT applications even in resource-constrained settings. Overall, the architecture supports industrial digital transformation efforts by improving operational efficiency, energy savings, and decision-making responsiveness in dynamic, data-intensive environments. The comparative study of the proposed model is given in Table 6.

*Table 6: Comparative study of the proposed model*

| Author name and Reference | Approach used | Response time |
|---|---|---|
| Lan et al. [21] | Universal Edge-Based Complex Event Processing Mechanism | 268.2s |
| Xu et al. [23] | (BalCon) and BalConPlus migration strategy | 370s |
| Priyadarsini et al. [24] | SALB | - |
| Proposed | HPC_SOA | 81s |

### 4.5 Strengths and Limitations of the Study

The proposed study presents a comprehensive solution that combines CEP, priority-aware scheduling, and hybrid meta-heuristic optimization. This integration is a key strength, addressing the shortcomings of fragmented solutions in previous works. The Hybrid Pigeon Cat Search Optimization Algorithm effectively balances computational load across fog nodes by combining global search (pigeon-inspired) and local refinement (cat-inspired) strategies. This reduces energy consumption and enhances resource utilization. Tasks are dynamically categorized based on urgency using multi-tier queues, which improves scheduling efficiency and minimizes unnecessary migration, as evident from performance metrics like reduced scheduling time and response delay.

The model outperforms state-of-the-art algorithms such as WOA, SMO, and GACO in key performance indicators including CPU usage, energy consumption, migration time, and number of migrated tasks, as shown in the comparative results. The system is designed to handle real-world scenarios using a publicly available Bitbrains dataset, ensuring relevance to industrial IoT and smart city applications. The integration of CEP, priority queues, and hybrid optimization increases system complexity, which might demand higher initial setup and computational overhead for smaller-scale deployments. Although effective, the CEP relies on predefined rules for identifying events. This could limit adaptability in environments with evolving event patterns unless periodically updated by domain experts.

## 5. CONCLUSION

The increasing demand for real-time analytics in IoT-driven environments has introduced complex challenges in task scheduling, resource optimization, and load balancing across edge and fog layers. This research was motivated by critical limitations identified in existing systems—namely, the lack of integrated models capable of handling dynamic workloads with contextual intelligence and efficient resource utilization. Specifically, the study aimed to answer how can an integrated, intelligent system improve task scheduling and load balancing in SDN-enabled fog computing environments while minimizing response time, energy consumption, and computational overhead? To address this, we proposed a novel framework that combines CEP for real-time task classification, a Multi-tier Priority Queue model for context-aware scheduling, and a HPC_SOA for intelligent load balancing. Through simulation on the Bitbrains dataset using iFogSim, the proposed approach demonstrated superior performance across key metrics scheduling time (8.123 sec), response time (81 sec), migration time (8.985 sec), energy consumption (22051 kWh), and CPU utilization (2723.06 MB) when compared with existing methods such as WOA, SMO, and GACO. The CEP module enhanced situational

responsiveness by identifying complex patterns from data streams, while the priority queue ensured timely execution of urgent tasks without causing starvation. HPC_SOA further optimized load balancing by adaptively distributing tasks across fog nodes based on real-time fitness evaluations considering energy, proximity, and resource availability.

This study significantly advances the current understanding of edge-level task processing and fog-level load balancing in SDNs. While previous works have individually addressed either Complex Event Processing or load balancing using static or mono-strategy optimizations, this research offers a holistic and adaptive framework that combines both. By integrating CEP for intelligent task categorization, a multi-tier queue for dynamic priority scheduling, and a HPC_SOA for optimal load distribution, the proposed model bridges critical gaps identified in the literature namely, the lack of real-time responsiveness, energy-aware processing, and end-to-end task orchestration. The simulation results on a real-world dataset further substantiate its superior performance in reducing response time, energy consumption, and migration overhead compared to existing models. Thus, the study not only provides a technically sound model but also sets a new direction for future research in scalable, intelligent, and adaptive fog computing within SDN-enabled IoT environments.

## REFERENCES

[1] Munirathinam, S. (2020). Industry 4.0: Industrial internet of things (IIOT), *In Advances in computers. Elsevier 117*(1), 129-164.

[2] Tian, Z., Shi, W., Wang, Y., Zhu, C., Du, X., Su, S., Sun, Y., Guizani, N. (2019). Real-time lateral movement detection based on evidence reasoning network for edge computing environment, *IEEE Transactions on Industrial Informatics 15*(7), 4285-94.

[3] Qi, Q., Tao, F. (2019). A smart manufacturing service system based on edge computing, fog computing, and cloud computing, *IEEE access 7*, 86769-77.

[4] Salaht, F.A., Desprez, F., Lebre, A. (2020). An overview of service placement problem in fog and edge computing, *ACM Computing Surveys (CSUR) 53*(3), 1-35.

[5] Du, X., Cardie, C. (2020). Event extraction by answering (almost) natural questions, arXiv preprint arXiv:2004.13625.

[6] Cortés-Ciriano, I., Lee, J.J., Xi, R., Jain, D., Jung, Y.L., Yang, L., Gordenin, D., Klimczak, L.J., Zhang, C.Z., Pellman, D.S. (2020). Comprehensive analysis of chromothripsis in 2,658 human cancers using whole-genome sequencing, *Nature genetics 52*(3), 331-41.

[7] Subramaniyaswamy, V., Manogaran, G., Logesh, R., Vijayakumar, V., Chilamkurti, N., Malathi, D., Senthilselvan, N. (2019). An ontology-driven personalized food recommendation in IoT-based healthcare system, *The Journal of Supercomputing 75*, 3184-216.

[8] Eskandari, M., Janjua, Z.H., Vecchio, M., Antonelli, F. (2020). Passban IDS: An intelligent anomaly-based intrusion detection system for IoT edge devices, *IEEE Internet of Things Journal 7*(8), 6882-97.

[9] Manzoor, S., Mazhar, F., Binaris, A., Hassan, M.U., Rasab, F., Mohamed, H.G. (2023). An Adaptive Symmetrical Load Balancing Scheme for Next Generation Wireless Networks, *Symmetry 15*(7), 1316.

[10] Moravejosharieh, A.H., Palmeira, F.C. (2019). Load-balancing In Software-Defined Networking: An Investigation on Influential System Parameters, *in 2019 29th International Telecommunication Networks and Applications Conference (ITNAC), IEEE* 1-6.

[11] Lu, J., Zhang, Z., Hu, T., Yi, P., Lan, J. (2019). A survey of controller placement problem in software-defined networking, *IEEE Access 7*, 24290-307.

[12] Urrea, C., Benítez, D. (2021). Software-defined networking solutions, architecture and controllers for the industrial internet of things: A review, *Sensors 21*(19), 6585.

[13] Mokhtar, H., Di, X., Zhou, Y., Hassan, A., Ma, Z., Musa, S. (2021). Multiple-level threshold load balancing in distributed SDN controllers, *Computer Networks 198*, 108369.

[14] Gawade, Y., Harale, A., Ekhe, D., Anjum, S., Lokhande, P. (2019). Video streaming over software defined networking with server load balancing, *Research Journal of Engineering Technology and Managemen*t (ISSN: 2582-0028) 2(01).

[15] Babbar, H., Rani, S. (2019). Emerging prospects and trends in software defined networking, *Journal of Computational and Theoretical Nanoscience 16*(10), 4236-41.

[16] Hongvanthong, S., Chunlin, L. (2022). A novel four-tier software-defined network architecture for scalable secure routing and load balancing, *International Journal of Communication Systems 35*(1), e5020.

[17] Li, S., Xin, Z., Xu, X., Zhang, Z. (2020). Load Balancing Algorithm of SDN Controller Based on Dynamic Threshold, *In 2023 3rd Asia-Pacific Conference on Communications Technology and Computer Science (ACCTCS), IEEE* 517-520.

[18] Sakthivel, M. (2021). An Analysis of Load Balancing Algorithm Using Software-Defined Network, *Turkish Journal of Computer and Mathematics Education (TURCOMAT) 12*(9), 578-86.

[19] Albowarab, M., Zakaria, N., Abidin, Z. (2019). Load balancing algorithms in software defined network, *International Journal of Technology and Engineering (IJRTE) 7*.

[20] Kofi, E.O., Ahene, E. (2023). Enhanced network load balancing technique for efficient performance in software defined network, *Plos one 18*(4), e0284176.

[21] Lan, L., Shi, R., Wang, B., Zhang, L., Jiang, N. (2019). A universal complex event processing mechanism based on edge computing for internet of things real-time monitoring, *IEEE Access 7*, 101865-78.

[22] da Costa Bezerra, S.F., Filho, A.S., Delicato, F.C., da Rocha, A.R. (2021). Processing complex events in fog-based internet of things systems for smart agriculture, *Sensors 21*(21), 7226.

[23] Xu, Y., Cello, M., Wang, I.C., Walid, A., Wilfong, G., Wen, C.H., Marchese, M., Chao, H.J. (2019). Dynamic switch migration in distributed software-defined networks to achieve controller load balance, *IEEE Journal on Selected Areas in Communications 37*(3), 515-29.

[24] Priyadarsini, M., Mukherjee, J.C., Bera, P., Kumar, S., Jakaria, A.H., Rahman, M.A. (2019). An adaptive load balancing scheme for software-defined network controllers, *Computer Networks 164*, 106918.

[25] Li, G., Cui, W., Liu, S., Zhao, W. (2022). A Load Balancing Strategy Based on Fuzzy Satisfaction Among Multiple Controllers in Software-Defined Networking, *Journal of Network and Systems Management 30*(3), 49.

[26] Kulshrestha, U., Durbha, S. (2020). Edge analytics and complex event processing for real time air pollution monitoring and control, *In IGARSS 2020-2020 IEEE International Geoscience and Remote Sensing Symposium, IEEE* 893-896.

[27] Liao, J.X., Wu, X.W. (2020). Resource allocation and task scheduling scheme in priority-based hierarchical edge computing system, *In 2020 19th international symposium on distributed computing and applications for business engineering and science (DCABES)*, *IEEE* 46-49.

[28] Cui, Z., Zhang, J., Wang, Y., Cao, Y., Cai, X., Zhang, W., and Chen, J. (2019). A pigeon-inspired optimization algorithm for many-objective optimization problems. *Science China, Information Sciences 62*(7), 70212.

[29] Dehghani, M., Hubálovský, Š., Trojovský, P. (2021). Cat and mouse based optimizer: A new nature-inspired optimization algorithm, *Sensors 21*(15), 5214.

[30] http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains