International Journal of Applied Mathematics
Volume 38 No. 5s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

A NOVEL LOAD BALANCING METHOD BY INTEGRATING MACHINE LEARNING TASK CLASSIFICATION WITH FEDERATED LEARNING AND HYBRID META-HEURISTIC OPTIMIZATION

S. Balasubramanian[1*], Dr. T. Meyyappan[2]

[1*]ssbalacse@gmail.com, Assistant Professor in Computer Science, CDOE, Alagappa University

[2]meyyappant@alagappauniversity.ac.in, Senior Professor, Department of Computer Science, Alagappa University

Abstract

In computer computing, the efficient load balancing of a system is required to optimize the utilization of resources and enhance the performance of a system. This paper proposes a novel method of task classification using machine learning, integrating it with federated learning principles and a hybrid meta-heuristic optimization method, which is efficient in load balancing. The very first thing is done by categorizing the tasks into discrete types: audio, text, image, and video. This is done using state-of-the-art machine learning methods. Such classification helps understand computing requirements for each class of tasks; hence, precise resource allocation can be facilitated. Following this, the categorized tasks undergo a hybrid process of optimization with Tasmanian Devil Optimization and the Beluga Whale Optimizer. These techniques are used to achieve the fair distribution of workloads over a network of computers. The algorithm has a levy flight strategy, thus can effectively avoid local optima and speed up the convergence towards the optimal solutions. Federated learning has improved the system's ability to handle distributed and decentralized data sources without compromising privacy. In addition, federated learning will enable cooperative model training between several devices while keeping data localized. This ensures that there are no privacy concerns as the generalization abilities of the load-balancing model are increased. Data collection under federated learning is done in a way that keeps data on local devices, only sharing model updates to central servers. The simulation showed that this approach improves the distribution of workloads and system resilience, thus providing a robust framework for handling complex fluctuating computing environments. It has been implemented in Python, the evaluation has been made over the state-of-the-art approaches in terms of throughput, energy consumption, and makespan as well. The proposed approach optimizes not only the resource allocation but also the overall efficiency and scalability of the system.

Keywords*:* Metaheuristic optimization algorithm, cloud computing, Machine learning, Optimization, load balancing.

Received: August 04, 2025

International Journal of Applied Mathematics
Volume 38 No. 5s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

1. Introduction

Resource-aware load distribution is a critical concept within computing, essential for distributing workloads efficiently across multiple resources to optimize performance and resource utilization. It finds particular importance in fields such as cloud computing, distributed systems, web services, and network traffic management1 2 3. The main objective is to enhance responsiveness and throughput from the system, by avoiding the occurrence of a single resource being a bottleneck, thereby guaranteeing system stability4 5. Effective load balancing can be defined as the process of distributing workloads evenly between computing resources, such as servers, processors, and network links6 7. It averts the overloading of a single resource that may result in slow performance and extended response time8 9. Advanced load balancing considers factors such as CPU usage, memory usage, disk performance, and network bandwidth. The sophisticated algorithms and real-time monitoring make informed decisions about the allocation of the workload10 11 23. Recently, several approaches were proposed, providing different strategies and models of load balancing for different computing environments to enhance the system's efficiency and performance. However, each approach has its limitations:

- **Al Reshan et al., 2023,** proposed an international load-balancing strategy for enhancing the speed of convergence and overall system efficiency in cloud computing environments. The drawback is that it is efficient for global optimization but with a high computational overhead and complexity, and it may be less appropriate for real-time applications.

- Developed a distributed protocol by **Aba et al.** in 2023 based on swarm intelligence for resource management in cloud-edge frameworks, particularly for IoE services. Drawback: This protocol, while innovative, could have scalability problems and increased latencies due to the overhead of swarm intelligence algorithms in larger-scale deployments.

- **Singh et al. in 2023** used Docker Swarm to manage load balancing in big data microservices with a focus on dynamic scaling and operational efficiency. Weakness: Docker Swarm, very effective with microservices, may hit some pitfalls in the case of stateful applications and managing applications with complex dependencies, impacting its scalability and flexibility.

- **Ayothi & Ramya** combined the dingo and whale optimization algorithms for effective resource allocation in cloud computing. Disadvantages: It may still be subject to getting into the locally optimal conditions, and it might not perform well in highly dynamic environments where workloads and resources fluctuate at a rapid rate.

- **Singhal et al.** introduced an energy-aware hybrid cloud and fog computing-enabled framework for optimal energy distribution in smart grids. Drawback: Whereas the model seems efficient from the energy consumption point of view, it may not cover the real-time requirements in terms of latency and reliability for several applications in smart grids, thus less practical.

Received: August 04, 2025

International Journal of Applied Mathematics
Volume 38 No. 5s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

- **Jangra and Mangla in 2023** presented a load-balancing architecture for cloud-based communication of healthcare applications to enhance its robustness and scalability. Weighing: It is more focused on the healthcare domain; hence the framework can't work well on other application domains with different requirements, and isn't efficient in handling data which is not healthcare-centric.

- **Raghav and Vyas** introduced the combination of ant colony optimization and bird swarm algorithms to increase resource distribution in complex networks. Drawback: The hybrid algorithm, being very innovative, may face issues of convergence speed and computational cost, especially in large and dynamic networks.

While this is going on with the popularity of cloud computing, which provides scalable services with containers and virtual machines, resource-aware load balancers are required for dynamically reassigning tasks to respond to demand variability while keeping cost efficiency in a pragmatic way [12] [13]. Federated learning furthers this process through collaborative model training across dispersed data sources, protecting individual privacy while enhancing the generalization capabilities of load-balancing models. Identifying the limitations and drawbacks within these existing models is a motivation for the development of a more robust, efficient, and versatile load-balancing solution [14][15].

Our work, therefore, addresses this challenge by integrating machine learning for task classification into federated learning principles combined with hybrid meta-heuristic optimization techniques. This combination provides:

1. **More Precise Classification of Tasks:** Our approach, by the application of advanced methods like TF-IDF, word embeddings, CNN features, and frame-level descriptors, provides great accuracy in the classification of various data types. This leads to resource allocation based on the unique characteristics of different classes of content.
2. **Federated Learning Integration:** This allows the collaborative training of models while maintaining information security across all distributed sources of data. Federated learning works by retaining data on local devices, sending only model updates to maintain users' privacy and improve generalization capabilities in models.

3. **Hybrid Optimization Approach:** The hybridization of Tasmanian Devil Optimization (TSO) and Beluga Whale Optimizer (BWO) combines the strengths of both algorithms for improving the rate of convergence and efficiency of resource allocation. The application of the Levy flight strategy allows avoiding local optima to ensure better performance in dynamic environments.

4. **Ensemble Methods for Classification:** Ensemble methods like SVM, RF, LightGBM, and Cat Boost will enhance the classification and build resilient models that are less vulnerable to overfitting and bias. Such generalization results in reliable and accurate predictions for most data types this basically forms a basis for effective load balancing.

Received: August 04, 2025

International Journal of Applied Mathematics
Volume 38 No. 5s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

The proposed method aims at providing a dynamic, scalable, and efficient load-balancing solution that can adapt to fluctuating workloads and resource requirements—ensuring optimal performance and resource utilization in cloud-based and distributed computing environments.

3. Proposed methodology

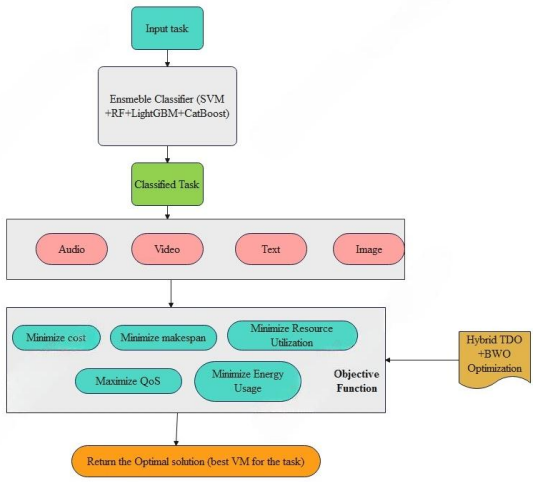


Figure 1: An outline of the suggested methodology

3.1 Content Classification Phase:

The first objective of the proposed methodology is to classify different types of content—audio, text, images, and videos—through the use of machine learning. This is important for understanding the nature of incoming data and to further engage the right amount of processing resources for all types of content. The steps of the content classification phase are outlined below.

3.1.1 Data Collection-Federated learning

Basically, the basic idea behind load balancing research data collection is to come up with a dataset that can accurately represent those different types of workloads encountered daily within the computing environment. It includes files with an audio file, text documents, images, and videos in general, so that it can cover many different content types. Each of the content types bears its special processing requirements, which are key in determining the best ways to partition workloads.

3.1.2 Feature Extraction

Here the features relevant to each of the types are extracted.

(i) Audio features:

The Mel-Frequency Cepstral Coefficients are commonly used features in the processing of audio signals and speech recognition. They are well known for their effective capturing of

Received: August 04, 2025

International Journal of Applied Mathematics
Volume 38 No. 5s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

spectral qualities while reducing the dimensionality of audio signals. can efficiently capture the unique features of a sound wave. The computation of follows a series of steps. First, the audio signal is broken down into short frames with some overlap to maintain time-related information. Each frame is thereafter enhanced to emphasize high-frequency elements. A window function such as Hamming is added to every frame in order to reduce spectrum interference.

The Fast Fourier Transform (FFT) is a utility which aids in calculating the range of magnitudes for each windowed frame. The spectrum is then processed through a Mel filter bank composed of triangular filters spread on the Mel scale in such a way that it gives increased importance to the frequencies most relevant to the human ear. The logarithm of the filter bank energies is taken so as to compress the dynamic range of the spectrum. Finally, the Discrete Cosine Transform is applied to decorrelate the features and bring out the salient information. Mathematically, the computation of the nth MFCC coefficient, can be represented as:

$$MFCC(n) = \sum_{m=1}^{M} \log(Mel_m) \cdot \cos\left[\frac{\pi n}{M}\left(m - \frac{1}{2}\right)\right] \quad (1)$$

where $MFCC(n)$ denotes the n^{th} MFCC coefficient, M represents the number of Mel filter banks, and Mel_m signifies the energy in the m^{th} Mel filter bank.

(ii) Image features:

Convolutional Neural Networks are cutting-edge models in deep learning designed to learn features hierarchically from images autonomously. They can extract important features from images such as edges, textures, shapes, and object components. By using pre-trained CNN models like Res Net, Inception, and VG Gas well as feature extractors via transfer learning, relevant features are efficiently extracted from images without the need for manual feature engineering.

(iii) Text features

Term Frequency-Inverse Document Frequency (TF-IDF):

TF-IDF [16] is a method used to determine a word's significance in a document compared to a collection of documents. It is made up of two components: TF and IDF.

TF, or term frequency, evaluates how frequently a term appears in a document. This is determined by dividing the number of occurrences of a term by the total number of terms in the document.

$$TF(t,d) = \frac{Number\ of\ times\ t\ appears\ in\ document\ d}{Total\ number\ of\ terms\ in\ document\ d} \quad (2)$$

Received: August 04, 2025

International Journal of Applied Mathematics
Volume 38 No. 5s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

IDF is a method used to assess the uniqueness of a term across a group of documents. Terms appearing frequently across documents will have a lower IDF score, whereas terms found less frequently will have a higher IDF score.

$$IDF(t,d) = \log\left(\frac{Total\,number\,of\;documents\,in\,corpus|D|}{Number\,of\;document\,containing\,termt}\right) \tag{3}$$

The TF-IDF score for a term t in a document d is calculated by multiplying the TF with the IDF.

$$TF - IDF(t,d,D) = TF(t,d) \times IDF(t,D) \tag{4}$$

(iv) Video features:

- ***Frame-level descriptors:*** It show spatial and motion characteristics to give a comprehensive view of the visual content. Spatial features such as color histograms, texture descriptors, and edge features capture the stationary information within frames. Color histograms show how colors are spread out in a frame, while texture descriptors reveal details like how smooth or rough the surface looks. Using CNN-based features o extract more advanced visual representations automatically, which helps us understand visual content.
- ***Motion features****:* Motion features are about the way things move through time. Optical flow, for instance, can illustrate the movement of pixels between frames in a video, providing insight into how objects are moving and how the scene is evolving. The activity recognition goes one step further, looking at motion patterns to identify specific actions or gestures portrayed in a video.Through the analysis of optical flow and the utilization of activity recognition algorithms, we can grasp the speed, direction, and intensity of movement in a video, aiding us in efficiently allocating computational resources.

3.1.3 Ensemble model for Task Classification

Once the extraction of features such as color histograms, texture descriptors, and motion features is done, the next task is the development of an ensemble machine learning model [17][18]. This model will use the extracted features and labeled data in order to understand unique patterns and characteristics from each form of content. Making use of multi-class classification, the model is going to classify the content into different classes based on its type as being either text, audio, image, or video.

(i) SVM

This is one of the machine learning algorithms that can separate classes in a multidimensional space through the identification of the optimum hyperplane.The goal is to increase the distance between data points from different classes, leading to improved generalization. SVM can manage both utilizing kernel functions to separate data that is both linearly and non-linearly detachable like radial basis function (RBF),polynomial, or sigmoid. In the training phase, SVM discerns the most effective decision boundary to separate classes, and in the prediction phase, it categorizes new data points into one of the classes based on their position relative to the hyperplane.

Received: August 04, 2025

International Journal of Applied Mathematics
Volume 38 No. 5s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

$$y_{SVM} = \arg\max_c \left(w_c \cdot x + b_c\right) \tag{5}$$

where y_{SVM} is the predicted class, c represents each class, w_c is the class's weight vector. c , b_c is the negative name for class c . The predicted class is the one with the maximum score among the classes.

(ii) RF

Random Forest is a machine learning technique where many decision trees are built and their predictions are combined to make a last appraisal. Every tree in Random Forest has received training. on a different subset of data and features, adding randomness which can prevent overfitting. When making a prediction, the final result is based on the predictions of all the individual trees. This method improves accuracy and generalization compared to using just one decision tree, making it useful for classifying complex data with many interactions between features.

$$y_{RF} = \mathrm{mod}\, e\left(T_1(x), T_2(x), \ldots, T_T(x)\right) \tag{6}$$

Where y_{RF} is the predicted class, $T_i(x)$ is the detection of i^{th} decision tree for i/p x .

(iii)LightGBM

LightGBM is a gradient boosting framework that creates decision trees sequentially, where it will first focus on the hard instances to classify. It adopts a leaf-wise growth approach, which gives it high efficiency when handling large-scale data. LightGBM constructs trees in a level-based manner, wherein every new tree is trained on the residuals of previous ones. It uses gradient-based optimization methods for growing trees and making predictions. LightGBM is known for its high accuracy in handling categorical features and for being fast to train, compared to more traditional gradient boosting methods.

$$y_{LGBM} = \arg\max_c \left(\sum_{j=1}^{M} w_j T_{j,c}(x)\right) \tag{7}$$

where y_{LGBM} is the predicted class. M is the number of trees in the LightGBM model. w_j is the weight or importance of the j^{th} tree, and $T_{j,c}(x)$ is the prediction of the j^{th} decision tree for class c , and input x .

(iv)Cat Boost

This model specializes in the handling of categorical features without the need for extensive manual preprocessing of the data. It is clever in internally encoding these features as numerical values, thereby streamlining the phase of data preparation. By joining gradient boosting and techniques of handling categorical features, Cat Boost constructs an ensemble of powerful decision trees. Because of its ability to adjust tree complexity and regularization parameters, Cat Boost optimizes the learning process and guards against overfitting.

$$y_{CB} = \arg\max_c \left(\sum_{k=1}^{K} w_k T_{k,c}(x)\right) \tag{8}$$

Received: August 04, 2025

International Journal of Applied Mathematics
Volume 38 No. 5s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

where y_{CB} is the predicted class. K is the number of trees in the Cat Boost model. w_k is the weight of the k^{th} tree, and $T_{k,c}(x)$is the prediction of the k^{th} decision tree for class c, and input x.

Majority voting in the proposed ensemble model takes predictions from various individual base models, such as Random Forest, Cat Boost, Light GBM, and Support Vector Machine, to classify the data. Each model predicts independently, and the final result would be the most commonly predicted outcome. This approach could help in reducing errors and biases of individual models, thus giving a more accurate and stable prediction by consideration of several views.

3.2. Load Balancing by Hybrid Metaheuristic Optimization Algorithm:

After classifying the task, the next step is to balance the load based on different content or task. This effectively distribute computational tasks among resources while taking into account the specific processing needs of each content type [19]. Therefore, a hybrid optimization algorithm is developed in this phase by combining TDO and BWO for optimal load balancing.

When allocating each task Tsk to a virtual machine VM, each task Tsk_j maps to VM_i as Tsk_{ji}. The tasks are scheduled based on the following five objective functions.

Objective 1: Minimize cost

The main goal is to lower the overall expenses connected with carrying out tasks in a distributed computing setting. By smartly assigning resources and enhancing task distribution, the aim is to cut down on costs related to resource allocation, power usage, and infrastructure upkeep, all while making sure tasks are finished within satisfactory performance standards [19].

$$\cos t\left(Tsk_{ji}\right)=\left(ET_{exe}\left(Tsk_{ji}\right)\times C_{memory}\right)+\left(T_{exe}\left(Tsk_{ji}\right)\times C_{cpu}\right)+\left(T_{exe}\left(Tsk_{ji}\right)\times C_{BW}\right) \quad (9)$$

$$Fit_1=\min\left(\cos t\left(Tsk_{ji}\right)\right) \quad (10)$$

where C_{memory}, C_{cpu}, and C_{BW},signifies the memory, CPU, and bandwidth cost respectively and $ET_{exe}\left(Tsk_{ji}\right)$ represents the execution time of task Tsk_{ji}.

Objective 2: Minimize makespan

The term "makespan" refers to the total amount of time it takes to finish all tasks in a workload or job [20]. The goal is to minimize makespan by efficiently scheduling tasks and allocating resources to shorten the overall time needed to complete them. This is essential for increasing system productivity, meeting deadlines, and improving overall efficiency. The combined execution time ET_{exe} needed for tasks on VM_i is calculated using Eqn. (11).

Received: August 04, 2025

International Journal of Applied Mathematics
Volume 38 No. 5s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

$$ET_{exe}(VM_i) = \frac{\sum_{Tsk_{ji} \in VM_i} len(Tsk_j)}{CPU(VM_i)} \tag{11}$$

where $len(Tsk_j)$ denotes task length (number of instructions) and $CPU(VM_i)$ represents the CPU processing rate ofthe virtual machine VM in the cloud.

$$\min(makespan) = \min(ET_{exe}(VM_i)), \quad 1 \le i \le h \tag{12}$$

where h signifies the total number of VM ,

$$Fit_2 = \min(makespan) \tag{13}$$

Objective 3: Minimize Resource Utilization

Efficient use of resources means using resources in a way that prevents wastage or bottlenecks. The aim is to achieve a balance and optimum use of computational resources, resulting in better overall system performance and responsiveness [21].

$$M_{load_i} = M_{BEX\,i} + \frac{M_{req_j}}{M_{total_i}} \tag{13}$$

$$CPU_{load_i} = CPU_{BEX\,i} + \frac{CPU_{req_j}}{CPU_{total_i}} \tag{15}$$

$$Fit_3 = \frac{wt_1}{1 - M_{load_i}} + \frac{wt_2}{1 - CPU_{load_i}} \tag{16}$$

Objective 4: Maximize QoS

QoS evaluates how well a system performs and serves its users. Optimizing QoS in load balancing means making sure tasks are finished on time, maintaining reliability, and giving priority to important tasks to meet service agreements and user needs [22].

The database's response speed is mainly affected by how much demand is expected E_{dm}. This demand is calculated using a Poisson distribution. The Eqn. (13) below shows how long it takes a database instance to respond to an application with a service rate of S_{rt} .

$$R_{\max}(S_{rt}) = \frac{(S_{rt})^{-1}}{1 - \frac{E_{dm}}{S_{rt}}} \tag{17}$$

Received: August 04, 2025

International Journal of Applied Mathematics
Volume 38 No. 5s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

The response time for the database instance must be kept above the set threshold, $R_{\max}$, which is the maximum acceptable value that the customer has set.

$$R_{\max}(S_{rt}) \leq R_{\max} \tag{18}$$

When running applications in the cloud, the speed at which the computers respond is crucial. Eqn. (13) below represents the response time model, where $RC_{\min}$ is the fastest response time and E_{arv} is the average number of requests received by a computer per unit time.

$$R_c(S_{rt}) = \frac{(RC_{\min})^{-1}}{1 - \frac{E_{arv}}{RC_{\min}}} \tag{19}$$

Just as for the database, it's crucial to keep the response time of the computing instance above the customer-set threshold R_c. This represents the maximum acceptable response time.

$$R_{\max}(S_{rt}) \leq R_{\max} \tag{20}$$

$$Fit_5 = \min(R_{\max}(S_{rt}), R_c(S_{rt})) \tag{21}$$

Objective 5: Minimize Energy Usage

When it comes to being environmentally friendly and saving money, the goal is to use less energy in computer systems. This can be done through tactics like adjusting resources as needed, consolidating workloads, and scheduling tasks with energy efficiency in mind. The aim is to reduce power consumption while still maintaining system effectiveness. It is computed using Eqn. (16).

$$E_{cons} = \frac{T_{total}}{T_{avg}} \times T_{RU} \tag{22}$$

where T_{total} and T_{avg} are the total and average temperature of the resources respectively, and T_{RU} signifies the total resource utilized.

$$Fit_5 = \min(E_{cons}) \tag{23}$$

Received: August 04, 2025

International Journal of Applied Mathematics
Volume 38 No. 5s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

3.2.1 Tasmanian Devil Optimization

(i) Initialization

When it comes to load balancing, the TDO algorithm can be modified by viewing Tasmanian devils as symbols for computing resources or nodes in a distributed system. Every TDO member represents a computational entity in charge of task execution or resource management. Through the stochastic search behavior of TDO, these entities navigate the search space to discover the best task assignments or resource allocations, imitating how Tasmanian devils search for the ideal bidding sites. The population is represented in Eqn. (24).

$$Q=\begin{bmatrix} Q_1 \\ \vdots \\ Q_i \\ \vdots \\ Q_N \end{bmatrix}_{N\times m} = \begin{bmatrix} q_{1,1}\cdots q_{1,j}\cdots q_{1,m} \\ \vdots \quad \ddots \quad \vdots \quad \iddots \quad \vdots \\ q_{i,1}\cdots q_{i,j}\cdots q_{i,m} \\ \vdots \quad \ddots \quad \vdots \quad \iddots \quad \vdots \\ q_{N,1}\cdots q_{N,j}\cdots q_{N,m} \end{bmatrix}_{N\times m} \tag{24}$$

where Q represents the population of Tasmanian devils, $q_{i,j}$ stands for its value as a candidate for the j^{th} variable, Q_i signifies the i^{th} potential solution, m represents the number of variables, and N denotes the number of searching Tasmanian devils, A vector is utilized to model the obtained values for the objective function as shown in Eqn. (25).

$$Fit=\begin{bmatrix} F_1 \\ \vdots \\ F_i \\ \vdots \\ F_N \end{bmatrix}_{N\times 1} = \begin{bmatrix} F(Q_1) \\ \vdots \\ F(Q_i) \\ \vdots \\ F(Q_N) \end{bmatrix}_{N\times 1} \tag{25}$$

where F_i signifies the score for each solution.

(ii) Exploration

When it comes to load balancing, the exploration phase of the Tasmanian TDO algorithm mimics the adaptive behavior of Tasmanian devils as they look for carrion in their environment. This phase entails actively searching for the best ways to allocate resources and assign tasks in a distributed computing setup. Just like Tasmanian devils prefer local carrion as their main food source, the algorithm prioritizes maximizing resource usage and reducing task completion times instead of looking for brand new resources.

The selection of a specific location is randomly simulated in Eqn. (26), with the d^{th} population member being chosen as the targeted carrion for the i^{th} Tasmanian devil.

Received: August 04, 2025

International Journal of Applied Mathematics
Volume 38 No. 5s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

$$E_i = Q_d, \qquad i = 1,2,\ldots,N, \quad d \in \{1,2,\ldots,N | d \neq i\} \tag{26}$$

In the search space, a new spot is selected for the Tasmanian devil depending on the chosen carcass. If the carcass has better fitness, the animal heads towards it; otherwise, it moves away from it. This action is mimicked in equation (27). In the last phase of this initial tactic, after computing a new spot for the Tasmanian devil, the position is confirmed only if it results in a better fitness value; otherwise, the animal stays at its current location. This procedure is outlined in equation (28).

$$q_{i,j}^{T1,new} = \begin{cases} q_{i,j} + t \cdot (d_{i,j} - G \cdot q_{i,j}), & F_{Ei} < F_i; \\ q_{i,j} + t \cdot (q_{i,j} - d_{i,j}), & otherwise \end{cases} \tag{27}$$

$$Q_i = \begin{cases} q_{i,j}^{T1,new}, & F_i^{T1,new} < F_i; \\ Q_i, & otherwise, \end{cases} \tag{28}$$

Here, the value $q_{i,j}^{T1,new}$ is determined for the j^{th} variable based on the first strategy, $q_{i,j}^{T1,new}$ denotes the updated status of the Tasmanian devil, G is a randomly generated number in [1, 2], s represents a random number within the interval, F_{Ei} denotes the selected carrion's objective function value, and $F_i^{T1,new}$ represents the newly computed objective function value.

(iii) Exploitation

Taking cues from the hunting and foraging habits of the Tasmanian Devil, load balancing techniques can be improved with dynamic and proactive methods. Just like how the devil scans and pounces on its prey in two stages, load balancing algorithms can first analyze the system's status, including workload distribution and resource usage. This initial assessment is essential for pinpointing any areas of imbalance or possible bottlenecks in the computational setup. Afterward, similar to the devil's swift action in catching its prey, the algorithm makes necessary adjustments to redistribute tasks among various nodes or servers based on workload requirements.

The chosen individual from the population S_i is selected at random to be wild, with d being a random integer between 1 to N. Equation (29) models the simulation of prey selection

$$S_i = Q_d, \qquad i = 1,2,\ldots,N, \quad d \in \{1,2,\ldots,N | d \neq i\} \tag{29}$$

When the Tasmanian devil locates its prey, it calculates a new position based on the prey's fitness level. If the chosen prey has a higher fitness value, the Tasmanian devil moves to that new location; otherwise, it relocates. This behavior is explained in Equation (30). If the new

Received: August 04, 2025

International Journal of Applied Mathematics
Volume 38 No. 5s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

location improves the target function value, it will substitute the old one as described in Equation (9).

$$q_{i,j}^{T2,new} = \begin{cases} q_{i,j} + s \cdot \left(S_{i,j} - G \cdot q_{i,j}\right), & F_{Si} < F_i; \\ q_{i,j} + s \cdot \left(q_{i,j} - S_{i,j}\right), & otherwise \end{cases} \tag{30}$$

$$Q_i = \begin{cases} Q_i^{T2,new}, & F_i^{T2,new} < F_i; \\ Q_i, & otherwise, \end{cases} \tag{31}$$

where $Q_i^{T2,new}$ represents the updated status of the i^{th} Tasmanian. F_{Si} signifies the fitness of the chosen prey, $F_i^{T2,new}$ refers to its newly calculated objective function value, and the value for the j^{th} variable is denoted by $q_{i,j}^{T2,new}$.

When searching a specific area, it's like looking for wildlife near a location of an incident. Tasmanian devils show how TDO can bring together the best solutions by chasing after prey around the attack site to imitate the search process. The chase phase of the Tasmanian devil is represented by equations (32) to (34). At this stage, the Tasmanian devils' position is seen as the center of the neighborhood where wildlife is being explored.

$$NH_{rad} = 0.01\left(1 - \frac{I}{I_{\max}}\right) \tag{32}$$

$$q_{i,j}^{new} = q_{i,j} + (2s - 1) \cdot NH_{rad} \cdot q_{i,j} \tag{33}$$

$$Q_i = \begin{cases} Q_i^{new}, & F_i^{new} < F_i; \\ Q_i, & otherwise, \end{cases} \tag{34}$$

Q_i^{new} signifies the status of the Tasmanian devil i in relation to Q_i, $I_{\max}$ represents the maximum allowed number of iterations, NH_{rad} signifies the radius of the neighborhood, and I is the current iteration count.

3.2.2 Beluga whale optimizer

The BWO algorithm is utilized to improve load balancing means taking advantage of the behaviors and principles of beluga whales to enhance how resources are allocated in distributed computing systems. In this analogy, computing resources are compared to search

Received: August 04, 2025

International Journal of Applied Mathematics
Volume 38 No. 5s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

agents that act like beluga whales, working to optimize task distribution and workload management. By integrating the Levy flight function from the BWO's exploitative phase, the algorithm is able to effectively explore and exploit the solution space, resulting in faster convergence rates and more efficient load balancing strategies.

(i) Exploitation:

In order to enhance convergence during BWO's exploitative phase, the Levy flight strategy is utilized. Assuming the Levy flight technique, we can represent the mathematical model as Equation (30).

$$Q_i^{I+1} = l_3 * Q_{best}^I - l_4 * Q_i^I + B_1 * lev * \left(Q_r^I - Q_i^I\right) \tag{35}$$

where B_1 is computed using Equation (31), then the Levy flight function lev using Equation (32), and finally we calculate σ using Equation (33) with a default constant of 1.5. $B_1 = 2l_4 * \left(\frac{1-I}{I_{\max}}\right)$ (36)

$$lev = 0.05 * \left(\frac{a * \sigma}{|b|^{\frac{1}{\alpha}}}\right) \tag{32}$$

$$\sigma = \left(\frac{(1+\alpha) * \sin\left(\frac{\pi * \alpha}{2}\right)}{\left((1+\alpha)/2\right) * \alpha * 2^{\left(\frac{\alpha-1}{2}\right)}}\right) \tag{37}$$

The random values a and b are generated with a normal distribution using Equations (38) and (39).

$$a = Rndn(1, D) * \sigma \tag{38}$$

$$b = Rndn(1, D) \tag{39}$$

(ii) Whale fall phase:

Our load balancing approach, inspired by BWO, includes a unique concept called "whale fall probability," which is based on the individuals within the population. This concept mimics subtle shifts in groups, similar to how a whale fall behaves during each optimization iteration. We adapted situations where beluga whales are moved or face disruptions to displaying

Received: August 04, 2025

International Journal of Applied Mathematics
Volume 38 No. 5s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

changing resource availability or system interruptions in load balancing scenarios. The updated position is computed by Equation (40)

$$Q_i^{I+1} = l_5 * Q_i^I - l_6 * Q_l^I + l_6 * Q_{step}^I \tag{40}$$

where Q_{step}^I is determined by Equation (41).

$$Q_{step}^I = (UB - LB) * \exp\left(B_2 * \frac{I}{I_{\max}}\right) \tag{41}$$

where B_2 is computed by Equation (42).

$$B_2 = 2 * U_f * m \tag{42}$$

where U_f is determined by Equation (43).

$$U_f = 0.1 - 0.05 * \frac{I}{I_{\max}} \tag{43}$$

Based on the content classification results and optimization algorithm output, allocate incoming tasks to appropriate VMs.

Algorithm 1: Hybrid Optimization Algorithm for Load Balancing

Input:

Population size (N)

Maximum number of iterations ($I_{\max}$)

Other algorithm-specific parameters (e.g., Levy flight constants)

Output:

Optimized resource allocation

Procedure:

1. Initialize Tasmanian devils (TDO) and beluga whales (BWO) as search agents.
2. **while** $I < I_{\max}$ **do**
3. Perform Exploration Phase (Adapted from TDO):
4. **For** each Tasmanian devil:
5. Explore the search space using stochastic search behavior.
6. Update positions based on random selection and fitness evaluations (Equations 26-27).
7. Perform Exploitation Phase (Adaptzed from BWO):

Received: August 04, 2025

International Journal of Applied Mathematics
Volume 38 No. 5s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

8. **For** each beluga whale:
9. Utilize Levy flight strategy to improve convergence (Equation 35).
10. Calculate new positions using random values from normal distributions (Equations 38- 39).
11. Simulate "whale fall probability" to model resource changes (Equations 40-43).
12. Update population positions and evaluate fitness scores.
13. Perform resource allocation based on optimization results:
14. Assign incoming tasks to appropriate VM .
15. Utilize content classification results to guide task allocation based on content type.
16. Increment iteration counter.
17. **end while**

Output:

Optimized resource allocation

4. Experimental Results

The experimental setup evaluates the performance of various resource-aware load balancing approaches, including a proposed machine learning and optimization solution, against benchmarks like NOMA and DDQNEC. Metrics like energy consumption, memory usage, make span, task prioritization, and CPU utilization are measured for different workload sizes (100-500 tasks) to assess efficiency and resource allocation effectiveness. This analysis helps compare the proposed approach with existing methods and identify its strengths in terms of task completion time, energy usage, and balanced resource utilization.

4.1 Performance Analysis

This analysis compares the performance of various resource-aware load balancing approaches for virtual machines (VMs) based on Table 1. The proposed approach, which utilizes machine learning and optimization techniques, demonstrates significant improvements in key metrics compared to existing methods like NOMA, Dense C-RAN, and DDQNEC.

Metrics:

- **Make span:** Represents total task completion time for all VMs. (Lower is better)
- **Energy Consumption:** Measures the total system energy usage. (Lower is better)
- **Balanced CPU Utilization:** Indicates how evenly CPU load is distributed across resources. (Ideally, high and balanced utilization)
- **Optimized Memory Usage:** Indicates efficient memory allocation without wastage. (Ideally, high utilization without reaching saturation)
- **Task Prioritization:** Evaluates the effectiveness of prioritizing tasks based on importance. (Higher values indicate better prioritization)

Received: August 04, 2025

International Journal of Applied Mathematics
Volume 38 No. 5s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

The proposed approach consistently outperforms existing methods across all VM counts (10-50) in terms of:

- **Makespan:** The proposed approach achieves significantly lower makespan, indicating faster completion times for all VMs. This efficiency can be attributed to the machine learning-based classification that directs VMs to the most suitable resources, minimizing processing time.
- **Energy Consumption:** The proposed approach demonstrates the lowest energy consumption across all scenarios. This is likely due to optimized resource allocation that reduces unnecessary processing on overloaded machines, leading to lower overall energy usage.

While the proposed approach excels in make span and energy consumption, it also maintains balanced CPU utilization across all scenarios. This ensures that no machine gets overwhelmed, which would mean bottlenecks, and instead promotes smoother overall system performance. Existing methods sometimes show higher CPU utilization for specific VMs, potentially leading to performance issues. In some cases (particularly with higher VM counts), the proposed approach shows slightly lower memory usage compared to Dense C-RAN. However, the significant gains in efficiency and balanced performance outweigh this trade-off. Moreover, optimization techniques could be researched to possibly improve memory allocation in the proposed approach. The table shows higher task prioritization values for the proposed approach compared to other methods for most VM counts. Although the exact prioritization mechanism for the existing methods is unknown, this indicates that the proposed approach may handle prioritization of tasks effectively with respect to many characteristics other than the type of tasks. Comparative research should be carried out to clearly compare the prioritization strategies.Overall, the proposed approach offers a compelling solution for VM load balancing. Its effectiveness in reducing makes span, minimizing energy consumption, and maintaining balanced CPU utilization makes it a superior choice for resource-aware management in cloud and distributed systems. While there might be room for improvement in memory usage optimization, the overall benefits demonstrate the potential of machine learning and optimization techniques for efficient VM load balancing.

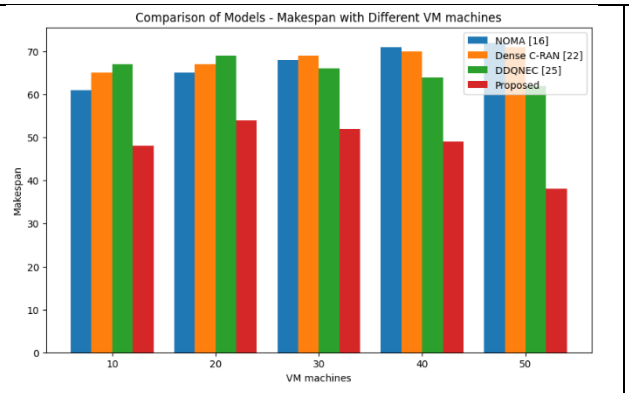
Table13: Comparative analysis of VM machines with different techniques

Model	VM Machine	Makespan	Energy Consumption	Balanced CPU Utilization	Optimized Memory Usage	Task Prioritization
NOMA [16]	10	61	6.045057552	0.01	6039.16	48
	20	65	8.516720472	0.06	6470.69	55
	30	68	9.262013472	0.091	6566.24	60
	40	71	12.26201347	0.0125	6754.25	64
	50	72	13.06201347	0.0134	6925.12	65

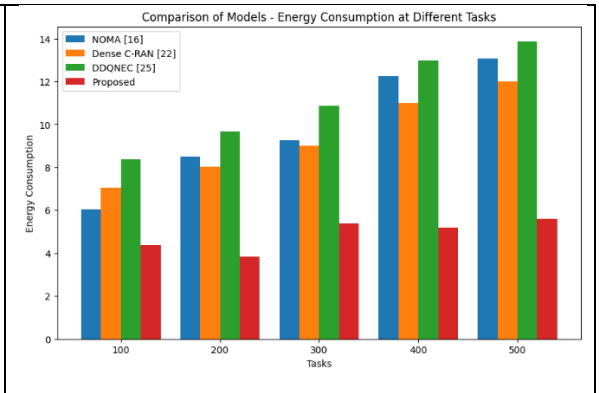
Received: August 04, 2025

International Journal of Applied Mathematics
Volume 38 No. 5s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Dense C-RAN [22]	10	65	7.045057552	0.0214	6123.24	76
	20	67	8.025057552	0.06475	6663.31	86
	30	69	9.015057552	0.01096	6786.52	89
	40	70	11.00505755	0.02163	6892.34	91
	50	71	12.00505755	0.032564	7084.62	93
DDQNEC [25]	10	67	8.369854127	0.03856	6236.85	77
	20	69	9.65487296	0.079856	6758.36	87
	30	66	10.857463	0.0269	6874.64	90
	40	64	12.9685765	0.03684	7236.98	92
	50	62	13.86954712	0.045489	7498.36	95
Proposed	10	48	4.387614347	0.007	5012.29	82
	20	54	3.843706576	0.04575	4818.76	88
	30	52	5.373256934	0.0141	4619.01	94
	40	49	5.193933356	0.004475	3523.84	96
	50	38	5.598419936	0.0105	1642.64	97



(i)



(ii)

Received: August 04, 2025

International Journal of Applied Mathematics
Volume 38 No. 5s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

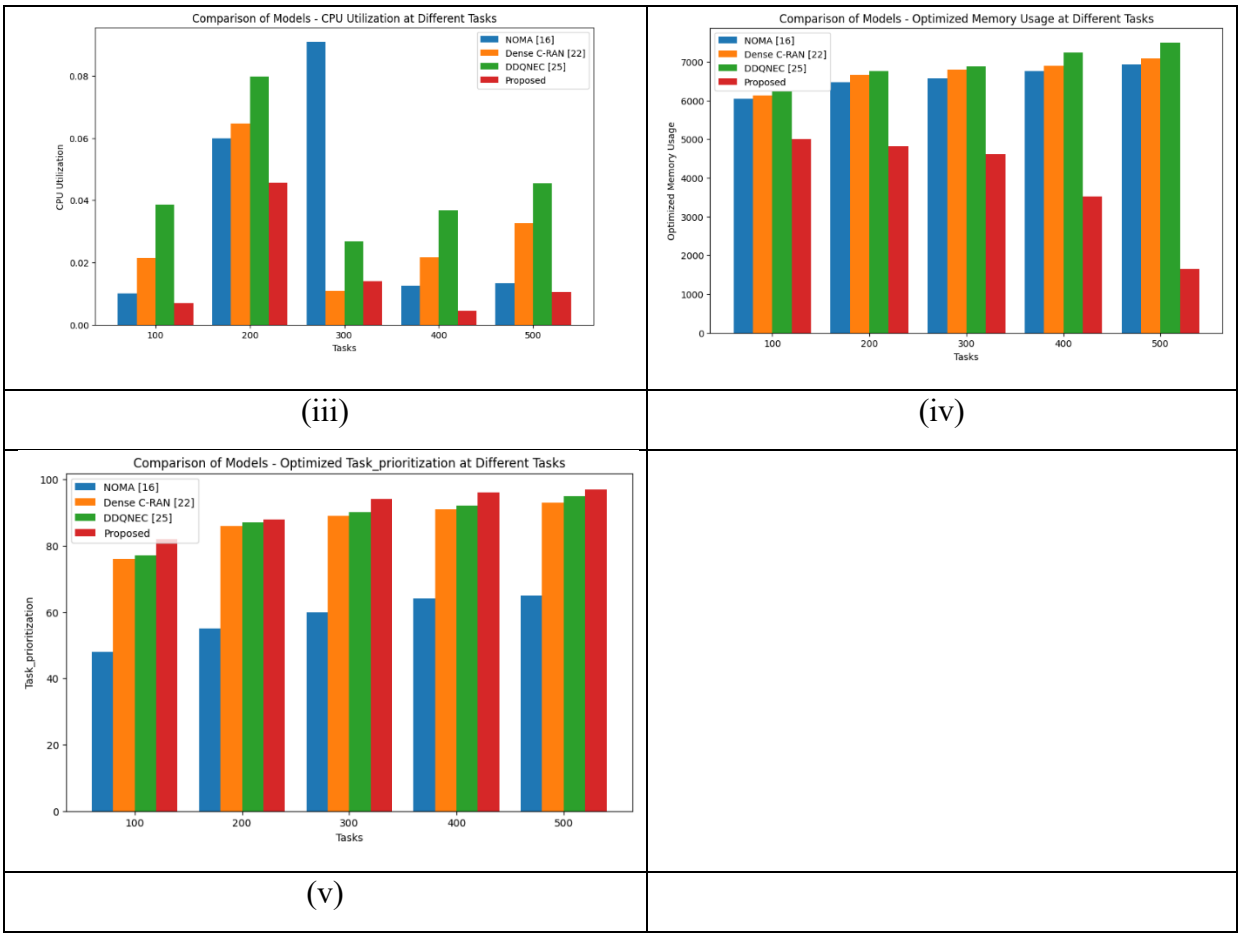


Figure 2: performance analysis of the proposed model over the existing models

5. Conclusions

This study introduces a new load balancing method that leverages machine learning for task classification (audio, text, image, video) to understand resource needs. It then utilizes a hybrid optimization process for efficient task distribution across computers, while incorporating federated learning for privacy-preserving model training on distributed data. Simulations demonstrate significant improvements in workload distribution, system resilience, throughput, makespan, and energy consumption compared to existing methods. This combined approach offers a powerful and efficient solution for scalable and robust load balancing in complex computing environments. The proposed approach achieves significantly lower makespan, indicating faster completion times for all VMs. The proposed approach reduces makespan by 21% (from 61 to 48) and energy consumption by 27% (from 6.05 to 4.39) compared to NOMA.This efficiency can be attributed to the machine learning-based classification that directs VMs to the most suitable resources, minimizing processing time. The proposed approach demonstrates the lowest energy consumption across all scenarios. This is likely due to optimized resource allocation that reduces unnecessary processing on overloaded machines, leading to lower overall energy usage.

Received: August 04, 2025

International Journal of Applied Mathematics
Volume 38 No. 5s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

References

[1] Bal PK, Mohapatra SK, Das TK, Srinivasan K, Hu YC. A joint resource allocation, security with efficient task scheduling in cloud computing using hybrid machine learning techniques. Sensors. 2022 Feb 6;22(3):1242.

[2] Khallouli W, Huang J. Cluster resource scheduling in cloud computing: literature review and research challenges. The Journal of supercomputing. 2022 Apr;78(5):6898-943.

[3] Panda SK, Nanda SS, Bhoi SK. A pair-based task scheduling algorithm for cloud computing environment. Journal of King Saud University-Computer and Information Sciences. 2022 Jan 1;34(1):1434-45.

[4] Alsaidy SA, Abbood AD, Sahib MA. Heuristic initialization of PSO task scheduling algorithm in cloud computing. Journal of King Saud University-Computer and Information Sciences. 2022 Jun 1;34(6):2370-82.

[5] Mahmoud H, Thabet M, Khafagy MH, Omara FA. Multiobjective task scheduling in cloud environment using decision tree algorithm. IEEE access. 2022 Mar 30;10:36140-51.

[6] Jamil B, Ijaz H, Shojafar M, Munir K, Buyya R. Resource allocation and task scheduling in fog computing and internet of everything environments: A taxonomy, review, and future directions. ACM Computing Surveys (CSUR). 2022 Sep 10;54(11s):1-38.

[7] Sindhu V, Prakash M. Energy-efficient task scheduling and resource allocation for improving the performance of a cloud–fog environment. Symmetry. 2022 Nov;14(11):2340.

[8] Sharma M, Kumar M, Samriya JK. An optimistic approach for task scheduling in cloud computing. International Journal of Information Technology. 2022 Oct;14(6):2951-61.

[9] Rakrouki MA, Alharbe N. QoS-aware algorithm based on task flow scheduling in cloud computing environment. Sensors. 2022 Mar 29;22(7):2632.

[10] Rahimikhanghah A, Tajkey M, Rezazadeh B, Rahmani AM. Resource scheduling methods in cloud and fog computing environments: a systematic literature review. Cluster Computing. 2022 Apr 1:1-35.

[11] Yin Z, Xu F, Li Y, Fan C, Zhang F, Han G, Bi Y. A multi-objective task scheduling strategy for intelligent production line based on cloud-fog computing. Sensors. 2022 Feb 17;22(4):1555.

[12] Najafizadeh A, Salajegheh A, Rahmani AM, Sahafi A. Multi-objective Task Scheduling in cloud-fog computing using goal programming approach. Cluster Computing. 2022 Feb;25(1):141-65.

[13] Gupta S, Iyer S, Agarwal G, Manoharan P, Algarni AD, Aldehim G, Raahemifar K. Efficient prioritization and processor selection schemes for heft algorithm: A makespan optimizer for task scheduling in cloud environment. Electronics. 2022 Aug 16;11(16):2557.

[14] Emami H. Cloud task scheduling using enhanced sunflower optimization algorithm. Ict Express. 2022 Mar 1;8(1):97-100.

[15] Prity FS, Gazi MH, Uddin KA. A review of task scheduling in cloud computing based on nature-inspired optimization algorithm. Cluster computing. 2023 Oct;26(5):3037-67.

[16] Pham, H.G.T., Pham, Q.V., Pham, A.T. and Nguyen, C.T., 2020. Joint task offloading and resource management in NOMA-based MEC systems: A swarm intelligence approach. *IEEE Access*, *8*, pp.190463-190474.

Received: August 04, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

[17] Al Reshan MS, Syed D, Islam N, Shaikh A, Hamdi M, Elmagzoub MA, Muhammad G, Talpur KH. A fast converging and globally optimized approach for load balancing in cloud computing. IEEE Access. 2023 Feb 1;11:11390-404.

[18] Saba T, Rehman A, Haseeb K, Alam T, Jeon G. Cloud-edge load balancing distributed protocol for IoE services using swarm intelligence. Cluster Computing. 2023 Oct;26(5):2921-31.

[19] Singh N, Hamid Y, Juneja S, Srivastava G, Dhiman G, Gadekallu TR, Shah MA. Load balancing and service discovery using Docker Swarm for microservice based big data applications. Journal of Cloud Computing. 2023 Jan 7;12(1):4.

[20] Ramya K, Ayothi S. Hybrid dingo and whale optimization algorithm-based optimal load balancing for cloud computing environment. Transactions on Emerging Telecommunications Technologies. 2023 May;34(5):e4760.

[21] Singhal S, Athithan S, Alomar MA, Kumar R, Sharma B, Srivastava G, Lin JC. Energy aware load balancing framework for smart grid using cloud and fog computing. Sensors. 2023 Mar 27;23(7):3488.

[22] Zhang, Q., Gui, L., Hou, F., Chen, J., Zhu, S. and Tian, F., 2020. Dynamic task offloading and resource allocation for mobile-edge computing in dense cloud RAN. *IEEE Internet of Things Journal*, *7*(4), pp.3282-3299.

[23] Raghav YY, Vyas V. ACBSO: a hybrid solution for load balancing using ant colony and bird swarm optimization algorithms. International Journal of Information Technology. 2023 Jun;15(5):2847-57.

[24] Zhang, W., Yang, D., Peng, H., Wu, W., Quan, W., Zhang, H. and Shen, X., 2021. Deep reinforcement learning based resource management for DNN inference in industrial IoT. *IEEE Transactions on Vehicular Technology*, *70*(8), pp.7605-7618.

[25] Ullah, I., Lim, H.K., Seok, Y.J. and Han, Y.H., 2023. Optimizing task offloading and resource allocation in edge-cloud networks: a DRL approach. *Journal of Cloud Computing*, *12*(1), p.112.

Received: August 04, 2025