ISSN: 2093-5374 / E-ISSN: 2093-5382

Efficient Load Balancing Using Enhanced Dragonfly with Firefly Optimization Over Cloud Computing Environment

P. Viswanatha Reddy[1*], and Dr.P. Savaridassan[2]

[1*]Research Scholar, Department of Networking & Communications, School of Computing, SRM Institute of Science and Technology, Kattankulathur, Tamil Nadu, India.
vp7891@srmist.edu.in, https://orcid.org/0009-0007-4694-1489

[2]Department of Networking and Communications, SRM Institute of Science and Technology, Kattankulathur, Tamil Nadu, India. savaridp@srmist.edu.in,
https://orcid.org/0000-0002-5088-3732

Received: February 04, 2026; Revised: March 12, 2026; Accepted: April 30, 2026; Published: June 30, 2026

Abstract

The large number of users of the cloud has resulted in a rapid increase in the number of requests for tasks, and scheduling tasks efficiently and load balancing them across non-homogeneous virtual machines (VMs) has become an urgent matter. This is a problem that is inherently NP-hard, especially when the Objective is to optimize multiple Objectives, such as makespan, execution cost, scheduling time, and resource utilization. Current meta-heuristic methods, such as the Particle Swarm Optimization (PSO), are computationally expensive and do not converge to global optimality when applied in isolation. To overcome these shortcomings, this paper proposes an Enhanced Dragonfly-Firefly Optimization (EDFO) algorithm, coupled with an Improved Advanced Encryption Standard (IAES) mechanism for safe and effective scheduling of cloud tasks. The EDFO model was an effort to merge the global search capability of the Dragonfly algorithm and the local refinement strength of the Firefly algorithm. The multi-objective function is formulated to minimize the maximum difference in VM completion times and to achieve even workloads. The results of the proposed EDFO-IAES are experimentally evaluated and shown to be much more effective than currently used methods such as DFGA and IBPSO-LBS. In particular, the imbalance level decreases to 0.45, the average storage usage increases to about 94%, and makepan is minimized by nearly a quarter across the workload variabilities. Besides, incorporating IAES increases data security without imposing a substantial computational burden. All in all, the suggested framework is more efficient, scalable, and secure, which is why it can be considered a viable solution for a real-life cloud computing setup.

Keywords: Cloud Computing, Load Balancing, Enhanced Dragonfly–Firefly Optimization (EDFO), Energy Efficiency, Improved Advanced Encryption Standard (IAES), Data Integrity and Security.

1 Introduction

Cloud computing can be considered an opportunity to share computing resources with other users. Designing some centers where programs are not tied to the hardware infrastructure and where resources

Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA), volume: 17, number: 2 (June-2026), pp. 49-72. DOI: *10.58346/JOWUA.2026.I2.004*

*Corresponding author: Research Scholar, Department of Networking & Communications, School of Computing, SRM Institute of Science and Technology, Kattankulathur, Tamil Nadu, India.

can be moved between virtual machines without affecting the task or source virtual machine is necessary for the development of datacenters (Alkhatib et al., 2021). Cloud computing is defined as the distributed, internet-based computing that enables users to access computing resources through a shared model. The challenges and issues surrounding the current state of cloud computing include resource allocation, load balancing, energy consumption management, task scheduling, etc. One key concern about cloud computing at the surface is load balancing. The dynamic allocation of tasks equally among two virtual machines in cloud computing will be achieved to avoid idle virtual machines, reduce the number of tasks, and create load balance between them (Mishra et al., 2020). Proper resource usage and load balancing between components may help decrease response time, increase fault tolerance and scalability, maximize user satisfaction, minimize heat and electricity consumption, reduce gas emissions, and minimize operational costs. Load balancing algorithms are available in two broad categories, i.e., 1) Static algorithms, 2) Dynamic algorithms (Sharma & Bedi, 2024). Static scheduling is a non-exclusive scheduling process in which activities are allocated to processors before the program commences. Scheduled decision-making is informed by the execution time of tasks, resource processing, etc.

In a cloud environment, the individual should be able to plan computational resources so that service providers can optimize resource use and users can execute their programs at the lowest cost. In fact, the large size of the functional programs, the heterogeneity and dynamism of resource features across virtual machines, and the presence of various requests within the cloud processing environment add complexity to achieving the required precision in predictions in this environment (Shah et al., 2024). In this context, over time estimation methods and optimization algorithms can be used to ensure progress in considering performance and efficiency in cloud processing networks. A cloud processing environment is used to schedule user requests based on available resources effectively. A good scheduling algorithm should provide a solution to some extent to failure-state occurrences so that such failures can be hidden from the user and the task can still be accomplished when conditions are good (Nehra & Kesswani, 2023). Scheduling algorithms are important enough to warrant consideration; thus, their relevance and intricacy warrant the importance of providing a method for optimizing the parameters, which could include load balancing, response time, execution time, task migrations, etc. Hence, the dragonfly optimization algorithm in this paper is used not only to distribute resources to tasks appropriately but also to set load balancing.

Among the significant factors that lead to energy inefficiency in the cloud computing setting is the non-uniformity of servers. Server idle time is high, and servers are often operating at 10-50% of peak capacity. This means that the servers are not catering to their respective optimal power-performance trade-offs, and that idle server mode consumes a large proportion of total power (Saboor et al., 2022). The mitigation of such values is achieved through the introduction of new cooling technologies and server and rack designs aligned with cloud computing infrastructure. The values, however, can also be reduced to very low levels in cloud data centers located in favourable geographical positions that allow them to benefit from ambient cooling (Mishra et al., 2024).

Nevertheless, cooling of cloud data centers through thermal means remains one of the major sources of energy inefficiency in cloud computing setups. Energy consumption is the most critical issue in content distribution systems and in most distributed systems (E.g., Cloud systems). These need an accumulation of connected computing resources by one or more providers in information centers spanning the globe. This is a censorious design parameter in existing data centers and cloud computing systems.

The benefits of cloud computing include data security, while the main considerations for widespread adoption are privacy, integrity, and trust. Cloud users require their sensitive information to be protected

against interference or hacking (Ghaseminya et al., 2025). The cloud computing platform is subject to internal and external security threats, as well as periodic outages and security incidents affecting cloud services. Data security concerns are critical to ensuring data integrity, privacy, and trust on the cloud platform. Cloud Computing presents new challenging security threats with inherent reasons underlying them.

To begin with, the traditional cryptographic knowledge-based protection is directly traceable due to the loss of control under cloud computing. Therefore, the confirmation of appropriate data storage in the cloud should be conducted without full knowledge of the data. With differing user data in the cloud and the need to ensure data security at all times, the question of proper data storage in the cloud is even more challenging (Nissar et al., 2024). Secondly, this type of Computing is not a third-party data warehouse. The information in the cloud is frequently modified by users, who can perform deletions, insertions, appends, modifications, reorders, etc. This is important to ensure correct, since dynamic data will need to be updated. Last but not least, Data centers work in parallel, collaborate, and operate in a distributed manner to enable Cloud Computing.

The mobility of steganography, compared to cryptography, is that the original secret message does not itself become an object of special focus. Thus, cryptography is the process of ensuring the security of message contents, not steganography, which concerns the secrecy of the existence and transmission of a secret message, or the safety of message contents. Security is high when steganography and encryption are used together. Encryption involves transforming plaintext into ciphertext. Steganography involves hiding information within a cover image (Pattnaik et al., 2022). Both encryption and steganography provide strong security, as encrypt the data, hide it within a cover image, and transmit it. The Objective of the research is to protect against security threats, including privacy, data integrity, and data tampering, and to ensure a secure environment. The work offers encryption of sensitive data, which frightens potential consumers and organizations from using cloud computing services for their sensitive data (Singamaneni et al., 2024). The paper will be beneficial and encourage researchers to conduct additional research on security solutions to support a reliable cloud environment. The proposed data security model will enhance the security of cloud user data.

Key Contributions and Novelty of the Work

The primary goal of this research is to create an effective and secure task scheduling system that not only minimizes makespan but also reduces load imbalance and enhances resource utilisation in a cloud computing system. To do this, a hybrid Enhanced Dragonfly-Firefly Optimization (EDFO) algorithm combined with an Improved Advanced Encryption Standard (IAES) mechanism is proposed.

The most important contributions of this work are as follows:

- An innovative hybrid EDFO algorithm that integrates the global exploration of Dragonfly and the local exploitation of Firefly, which is better with respect to convergence and optimal timing.
- A multi-objective optimization model that aims to reduce the degree of imbalance, make span, and execution cost, and maximize resource utilization.
- Embedding of IAES for secure delivery of task scheduling and data protection, without emphasizing a great amount of computation.
- Extensive experimental analysis demonstrates improved performance compared to existing methods, such as DFGA and IBPSO-LBS, across a range of workloads.

The originality of the proposed method lies in the successful hybridization of two meta-heuristic algorithms and a security-enhanced scheduling scheme, which together improve the efficiency of performance and data protection in the cloud environment.

The remainder of the research work is structured in the following way: Section 2 presents an overview of the latest methods in the area of load balancing, energy consumption modeling, and data security. Section 3 shows the proposed methodology which consists of a load-balancing algorithm, energy efficient model and data integrity mechanism. The results and discussion are given in section 4. The conclusion and future work are explained in section 5.

2 Literature Review

This section proves the topical research in the field of load balancing, energy consumption models, and data security. The methods and models proposed earlier by various authors to ensure data security in a compliant manner are limited.

Marri & Rajalakshmi, (2022), an article presents a Double-Fitness Genetic Algorithm (DFGA) for programming structure in Cloud Computing. This algorithm can be used to test the less desirable outcome of task scheduling, which alone results in a shorter overall task completion time but a shorter average completion time. A comparison and contrast of DFGA with the Adaptive Genetic Algorithm (AGA) is conducted in a simulation experiment, and the results indicate that DFGA is better and an efficient scheduling algorithm for tasks in a Cloud Computing environment. It was proposed that a load-balancing system with the use of soft computing can be used (Mandal et al., 2023). The local optimization algorithms include the stochastic hill-climbing, in which the incoming jobs are assigned to a server or a virtual machine (VM). Cloud Analyst is used in qualitative and quantitative the analysis of the performance of the algorithm. Cloud Analyst is a graphical Cloud computing and Cloud Sim environment modeller. It is also opposed to the Round Robin and First Come First Serve (FCFS). In order to prepare and enhance the significant actions of effective resource consumption as well as reaction time of the activities respectively, Devaraj et al., (2020) proposed a hybrid firefly and Improved Particle Swarm Optimization (IPSO) algorithms to derive the optimistic average load. Recommendations as to how the performance of the proposed hybrid method can be assessed have also been given in this paper. It was found that, with multiple objectives used, there were high performance on similar steps and flexible behavior in minimization of average load during optimization. This work will denote using the firefly algorithm developed (Karunya & Devi, 2025) as the firefly algorithm and the algorithm is suggested to be used in load balancing. To begin with, the index table will be kept depending upon the availability of virtual servers and order of request. The different formulae will then be used to determine the load index. The load balancing will be used in the firefly algorithm on the basis of load index. The analysis offered the presentation of the performance resulted in the expected outcomes and, therefore, acted as an indicator that the given strategy is effective to optimize schedules through the balancing of loads. The mean time of the completion of offered approach is 0.934 ms.

Masadeh et al., (2025) had generated a reasonable means of code and decoding individuals and determining that the overall energy capacity of the servers was the fitness value of that particular individual. Meanwhile, to speed up convergence and increase power of search, local search operator is presented. Lastly, the experiments indicate usefulness and effectiveness of the proposed algorithm. The resource scheduling algorithm based on the optimization method of saving energy had been suggested (Wang et al., 2023) in Cloud computing. The experimental outcomes show that hardware environment has the potential to save energy by lowering the energy usage of jobs which were not utilized to full capacity. Based on the frame of different power-saving policies of a system, Kondori & Peashdad,

(2015) developed the effective green control (EGC) algorithm in order to solve the limited optimization problems and tradeoffs among costs/performances. Also developed a cost function to take into account the utilization of power, the system congestion and server start up. All the effects of the energy-efficiency controls on the response times and operational modes and the amount of costs incurred are all depicted. It aims to determine the optimal service rate and mode-switching point in order to maximize the cost, at various arrival rates, and with a response-time guarantee. The outcomes of the simulation indicate that the advantages of the minimization of operating costs and maximization of response time may be proved by applying power-saving policies and suggested algorithm in comparison to a traditional system with the same performance guarantees.

To ensure data integrity on the untrusted server, Tian & Nogales, (2023) proposed a Merkle Hash Tree (MHT) and the AES algorithm. Most of the schemes that were proposed before ensured storage security using the RSA algorithm. Since AES is faster in encryption and decryption and requires a smaller buffer than RSA, seek to optimize the system using AES. The cloud will not allow users to validate the data store. Considering this and offloading client data integrity checking, propose a service provider, the Third-Party Auditor (TPA), that performs data integrity checks on behalf of the client and issues an alert indicating the status of the stored data. The storage security scheme also proposes a data recovery scheme as a warranty for the recovery of data in the event of loss or corruption. By doing so, the proposed scheme aims to maintain user data while supporting data restoration. The time the server would take to compute is also lower in the system than in previous systems.

In an effort to enhance the security of cloud storage, Akilashri & Ahadha Parveen, (2022) came up with a security model of Cloud Analyst which made use of various encryption algorithms and integrity validation scheme. And start with the storage selection phase that is broken down into three different parts that are: Private, Public and Hybrid. The three sections have embraced varying encryption procedures in accordance with security aspects like authentication, confidentiality, security, privacy, non-repudiation and integrity. The token-generation mechanism that is exceptional in the Privatized section is useful in providing user authentication. The Hybrid section also offers On-Demand Two-Tier security architecture and the public section offers light and fast encryption and decryption. All three parts check the integrity and encrypt overall data twice. The user would like to be prompted to insert their user login and password after which he would be granted access to the encrypted data stored in either the private, public or hybrid section. Hence, the hacker will find it hard to access the authorized environment. One of the proposals is a data security model of cloud computing developed based on separation of security at different layers presented in (Adee & Mouratidis, 2022). The given framework allows supporting the idea that the given strategy is able to enhance the safety of service-based systems and cloud computing applications. It was suggested in (Menaka & Kumar, 2024) that the binary implementation of the PSO algorithm was effective since it is low regarding time complexities, also it is cheap in the execution of the tasks scheduling and balancing in the cloud computing. Precisely, identify an intentional role that calculates the maximum time separation of accomplishment of individual VMs that should be implemented within the restricting and optimization constraints that are presented in this paper. Design A load-balancing strategy of a load-balancing strategy. Empirical algorithm the experimental findings show that the level of task scheduling and load balancing of the empirical algorithm is high compared to the existing metaheuristic and heuristic algorithm.

3 Proposed Methodology

The suggested algorithms in this paper are the Enhanced DragonflyFirefly Optimization (EDFO) and the Improved Advanced Encryption Standard (IAES) that incur low time complex and low cost in

scheduling and balancing cloud computing tasks. It is also better in terms of energy consumption and complexity of computation in cloud computing. Also, it disperses the load extensively within the machine. An upgraded version of Advanced encryption standard (AES) algorithm provides security. figure 1 illustrates the process of the proposed methodology of this study. The primary input that can be provided to the research work is as below: Cloud formation is used when the number of Virtual machines (VMs) and tasks are constant as well as the amount of resources, users and servers.

1. Energy-efficient task scheduling in cloud computing data centres. It is intended to minimize energy use without compromising job performance. It is also used to reduce task execution time.
2. Efficient load balancing of the cloud computing environment using enhanced Dragonfly-Firefly Optimization (EDFO) algorithm. It is intended to increase capacity (concurrent users) and the reliability of applications. It is also used to balance cumulative workloads within the cloud computing environment. It is applied to maximize resource utilization, throughput, and processing time in data centers, and to enhance user response time.
3. The Improved Advanced Encryption Standard (IAES) algorithm is used to ensure data integrity and security.

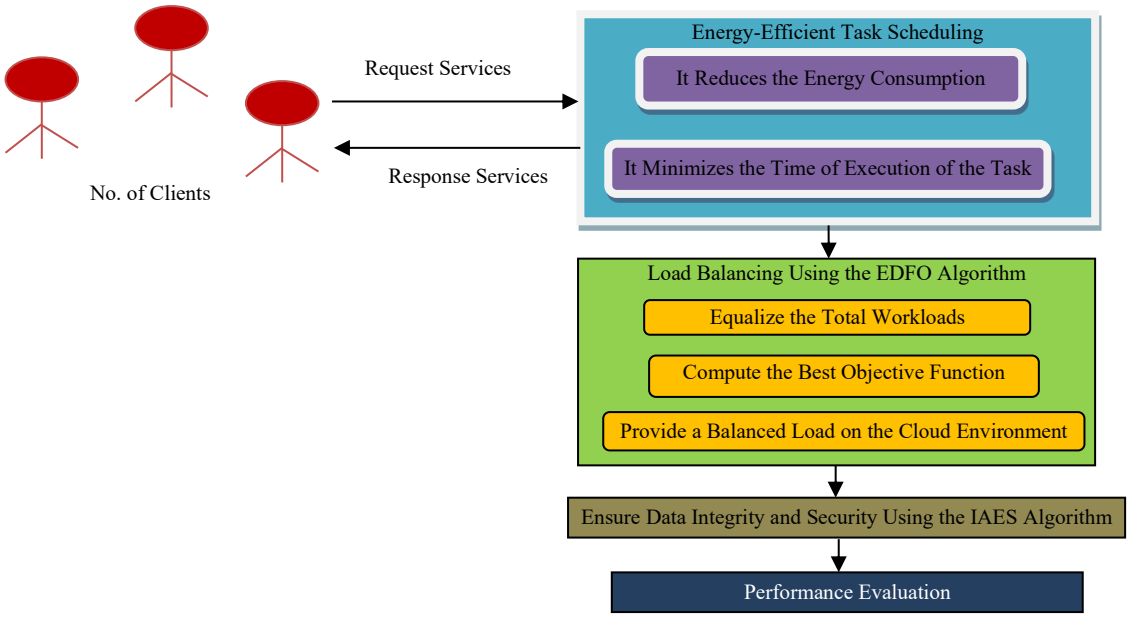

Figure 1: The process of the proposed methodology

3.1. Definitions

Definition 1: (Virtual Machine). The following Form can be defined as a virtual machine $V_M = \{id, mips, bw, pesnumber\}$ where id is the identifier number of a VM, mips (million instructions per second) is the execution speed of an individual processing element (PE) of a VM, bw is the bandwidth of a VM, and pesnumber is the number of PE in a VM (Menaka & Kumar, 2024).

Definition 2: (Task). One can define a task (T) as T = id, length, pes number with id being the identifier number of T, length being the size of T in million instructions (MI) and pes number being the number of PE that is required to execute a T on a suitable VM.

Definition 3: (Optimal Solution). The best solution in this paper is defined as the introduction of a heterogeneous task set on different heterogeneous VMs, which will decrease makespan, waiting time,

and imbalance, and increase resource utilization, decrease scheduling time (running time), and minimise the cost of execution.

Definition 4: (Degree of Imbalance). One value that is used to measure the level of load allocation among VMs in terms of their execution potential is the degree of imbalance (DI). The small DI means that it has a more balanced load. DI is calculated by equation (1).

$$DI = \frac{T_{max} - T_{min}}{T_{avg}} \quad (1)$$

Where T_{max}, T_{min}, and T_{avg} are the maximum execution time, minimum execution time, and average total execution time among all the VMs, respectively.

Definition 5: (Completion Time of a VM). The completion time of a VM is the time elapsed from the last task's completion. It is denoted by CT, as shown by equation (2) (Farajizadeh & Bakhsh, 2015).

$$CT_i = \sum_{j=1}^{n} \frac{T_j.length}{V_{mi}.pesnumber \times V_{mi}.mips}, with\ i \in \{1,2,\ldots,m\} \text{ and } j \in \{1,2,\ldots,n\} \quad (2)$$

Where m and n denote the number of VMs and tasks, respectively.

Definition 6: (Makespan). The total time required to complete all the tasks is known as the makespan. The fact that it has a low makespan is that the scheduler efficiently plans activities to resources. It is expressed by equation (3).

$$makespan = \max_{1 \le i \le m} \{CT_i\} \quad (3)$$

Where m denotes the number of VMs.

Definition 7: (Resource Utilization). Resource utilization is a performance indicator that measures resource usage at a high rate, demonstrating that the cloud provider can achieve maximum profit. Resource utilization (RU) is obtained using the following equation (4).

$$RU = \frac{\sum_{i=1}^{m} CT_i}{makespan \times m} \quad (4)$$

Definition 8: (Execution Cost). Execution cost (EC) refers to the charge, or the total value of compensation, made by a cloud user to a cloud provider for the use of resources to complete work. The overall goal of cloud users is to achieve cost reduction, efficient use, and minimum makespan. The execution cost is calculated using equation (5).

$$EC = \sum_{i=1}^{m} price_i * CT_i \quad (5)$$

3.2. Proposed Cloud System Model

In this case, the algorithm, the Enhanced Dragonfly-firefly Optimization (EDFO) algorithm, to solve the problem of optimal scheduling and balancing of the heterogeneous tasks on the heterogeneous virtual machines within a short time. To start with, present a proposal of a new algorithm in the Enhanced Dragonfly Optimization-Firefly Optimization (EDFO) algorithm on the task scheduling and load balancing problem in cloud computing in detail. And finally, introduce the given algorithm and approximate its time complexity.

The present work gives the specifications of a new EDFO, which is low complexity in time and low cost hence, the most suitable way to schedule and balance the heterogeneous user jobs on the heterogeneous virtual machines in the cloud. In particular, the suggested EDFO framework is to be introduced and the purposeful role of the issue is to be stipulated. The primary goal of the suggested methodology is to distribute all tasks to a VM based on its capacity (load). Load-balancing algorithms are applied to eliminate the tasks of the overloaded VM and migrate it to the load VM. The suggested

system will comprise a huge amount of data centers or physical machines (PM) and each physical machine will contain some virtual machines (VMs) to fulfill the tasks of the user. The users of the cloud have various tasks to fulfill in the VM. The operations in this work are assigned to VM through a load-balancing algorithm. The load-balancing code proposed is constantly monitoring load of each VM in the cloud. The load of the VM will depend on the amount of time, which is required to implement a task. The VM load is also varied, since it takes varying amount of time to execute one task and the other task.

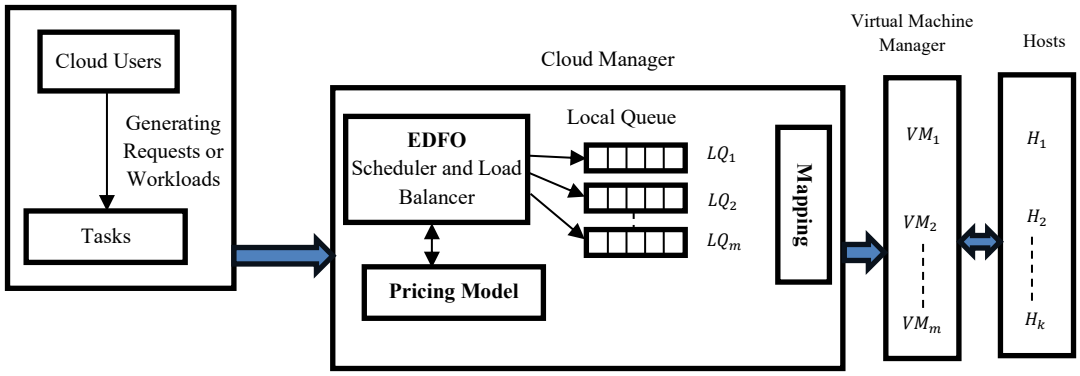

Figure 2: The process of load balancing and scheduling using EDFO

It is proposed that a common cloud computing model could help to identify the best possible solution to the task scheduling problem and load balancing with the assistance of EDFO, as shown in figure 2. The cloud system has the following 3 modules:

- The user request is the first module and is broken down into a large number of tasks and submitted to the cloud manager.
- The second module is the cloud manager (CM), which is the one that creates a task-local queue per VM. Two sub modules exist under CM, including: the proposed algorithm EDFO, the pricing model and the Mapping. All the activity between heterogeneous VMs is assigned and managed by EDFO according to update and optimization constraints and each local queue is assigned to the VMs by Mapping. In the meantime, the pricing model identifies the best possible cost of executing individual user tasks.
- The third module is the virtual machine manager which will depend on other hosts. It is also in charge of a bundle of virtual machines to run user workloads.

3.3. Objective Function

Considering m heterogeneous virtual machines (VMs) running on different hosts and executing n heterogeneous tasks, each VM_i has its completion time CT_i, as shown by equation (6). Among m heterogeneous VMs, define the completion time difference expressed as:

$$dct = \left|CT_i - CT_j\right| \quad (6)$$

Where i and j take the values 1, 2, 3, and m, and i is not equal to j. DCT aims to identify the overloaded and underloaded VMs. According to equation (6), the objective is to maximize the completion time difference across all heterogeneous VMs, aiming to reduce the imbalance between available heterogeneous VMs and the overall completion time and waiting time of user tasks within each VM. Accordingly, the objective function has the form given by equation (7).

$$f(CT) = max\{CT_i - CT_j|/1 \leq i|j \leq m\} \text{ where } CT = (CT_1, CT_2, \ldots., CT_m) dct_{max} = f(CT) \quad (7)$$

Subject to updating and optimization constraints, the calculations are based on a sigmoid function and the minimum completion time across various VMs.

3.4. Enhanced Dragonfly and Firefly Optimization Algorithm (EDFO)

The main purpose of the proposed methodology is to outsource the job to VM using EDFO and, therefore, achieve the reduction of the overall amount of execution time and cost and equal the load. Another significant parameter of scheduling is the load, which characterizes the process of allocating the load to the different nodes of a distributed system to make the most of the resources possible and enhance the responsiveness of the tasks without making another part of the VM overload. The other nodes are either idle or working at a very low rate. To avoid this issue, the multi-objective-based load balancing is maximized in the work with the assistance of EDFO. The Dragonfly Optimization Algorithm is a metaheuristic algorithm, which is inspired by the natural swarming behavior of the dragonfly not just in locomotion, but also in rest posture. Two types of swarm exist in dragonflies (static swarm) and migration (dynamic swarm). the dragonflies fly and cover a great distance and various places and this is why the mobile swarm has so many dragonflies swarming it that it is the exploration stage. More widely are the dragon-flies of the stationary swarm and of a swarm whose developments and changes of the flight direction are immediate, and this can be explained at the stage of exploitation. To boost performance in the searching and avoid local optima, this study will use a mixture of FA and DA. The general idea of the suggested strategy is shown in figure 2. The task scheduling of a multiobjective load-balancing based is provided step-by-step.

Step 1: Initialization

This is a random initiation of the population of dragonflies. The population is a group of solutions. The solution is created basing on the number of user tasks and virtual machines (VMs) (Hammouri et al., 2020). First, VMs are assigned tasks at random. Then, the solutions are updated according to the fitness function. The first solution is provided in equation (1). The solution size is the number of steps in the task, and each Dragonfly is an available node. Considering the example of 10 tasks and 5 nodes, a population is 10, and a dragonfly can predict to 1, 2, 3, 4, or 5. The encoding of the sample solution is provided in equation (8) below.

$$P_i = \{2,4,3,1,5,2,5,3,2,1\} \quad (8)$$

Based on the equation above, VM2 will be assigned task 1, VM4 will be assigned task 2, and VM1 will be assigned task 10. Table 1 shows the relationship between VMs and tasks. Based on the solution encoding, generate the population matrix (SM), whose elements are 0 or 1. The model is constructed based on the correlation between tasks and VMs. The probability that there is an associated Ness for task j with vm_i is either $PM(i,j) = 1$ or $PM(i,j) = 0$. Therefore, the number 1 in each column is one, and the rest of the columns have 0. The corresponding PM is, in equation (9).

$$PM = \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} \quad (9)$$

Step 2: Fitness Calculation

Once the solutions are produced, evaluate their fitness. The fitness function is provided in equation (10).

$$MOF = \min[\alpha_1(ET) + \alpha_2(EC) + \alpha_3(1 - load)] \quad (10)$$

Table 1: Connection between VM and task

Node	Corresponding Task
1	4, 10
2	1, 6, 9
3	8, 3
4	2
5	5, 7

Step 3: Update Using the Dragonfly Algorithm

This algorithm operates on five key factors: separation, alignment, cohesion, food attraction, and enemy avoidance. Separation is determined by the help of equation (11).

$$SP_i = -\sum_{j=1}^{N} X - X_j \quad (11)$$

X is the location of the current person, X_j is the location of the jth-nearest neighbouring person, and N is the size of the surrounding. The alignment is determined by using equation (12).

$$A_i = \frac{\sum_{j=1}^{N} V_j}{N} \quad (12)$$

Where V_j is the velocity of the jth nearest individual. The cohesion is computed by the use of equation (13).

$$C_i = \frac{\sum_{j=1}^{N} X_j}{N} - X \quad (13)$$

Where X is the position of the current individual, N is the neighbourhood, and X_j is the position of the jth neighbour individual. The attraction to a food source is computed in the following way, as shown in equation (14):

$$F_i = X^+ - X \quad (14)$$

In which X = the location of the current individual, and X^+ is the location of the food source. The formula determines the direction to an enemy, as shown in equation (15):

$$E_i = X^- - X \quad (15)$$

Where X is the position of a current person and X^- indicates the position of the enemy. Calculation of the velocity vector is done after position calculation, as given in equation (16).

$$\Delta X_{k+1} = (sS_i + aA_i + cC_i + fF_i + eE_i) + w\Delta X_i \quad (16)$$

Where s is the separation weight, S_i is the separation of the i-th individual, a is the alignment weight, A_i is the alignment of the i-th individual, c is the cohesion weight, C_i is the cohesion of the i-th individual, f is the food factor, F_i is the food source of the i-th individual, e is the enemy factor, E_i is the enemy position location of the i-th individual, w is the inertia weight and k is the iteration count. Once the step vector has been determined, the position vectors are determined in the following way, as shows in equation (17):

$$X_{k+1} = X_k + \Delta X_{k+1} \quad (17)$$

Where k is the current iteration.

Step 4: Update Using the Firefly Algorithm

First of all, randomly select the initial population according to the number of flies. Then, compute the fitness function using (11). The motion of the Firefly (FF), which is attracted to a different and more attractive firefly j, is based on (Yang & Slowik, 2020).

$$x_{i+1} = x_i + B_0 e^{-\gamma r^2}\left(x_j - x_i\right) + \varphi\left(rand - \frac{1}{2}\right) \quad (18)$$

The second term in equation (18) The reason is that (18) results from attraction; the third term introduces the randomization 0/0, and rand is a uniformly distributed random number between 0 and 1. In EDFO, the fitness function is computed, and the most appropriate solution is selected based on its fitness value.

Step 5: Select the Best Solution

When comparing DA and FA, when the best fitness value of the DA_{best} is smaller than the best fitness value of FA_{best}, the best position of DA is substituted by FA. Otherwise, the FA fitness value is lower than the DA fitness value; DA takes the position.

Step 6: Termination Criteria

The iteration process stops when an improved solution is found or when the given number of iterations is reached. The cloud is provided with the final solution.

3.4. Security Analysis Using Improved Advanced Encryption Standard (AES)

AES is an iterative cipher rather than a Feistel cipher. It is founded on two world encryption and decryption strategies of data which involve substitution and permutation network (SPN). SPN: This is a collection of mathematical functions that are carried out in block cipher functions. The plaintext block size that AES offers is 128 bits (16 bytes) non-randomizable. These 16 bytes are presented as a 4x4 matrix and AES process is carried out on the 4x4 block of the bytes. The other notable observation of AES is the number of rounds (Manoj Kumar & Karthigaikumar, 2020), other than that. The rounds are dependent on the length of the key. There are three major sizes of AES algorithm which include 128, 192 or 256 bits. Depending on key size, there are 10 rounds of AES using 128-bit keys, 12 rounds with key size 192 and 14 rounds with key size 256.

3.4.1. Encryption Process

Encryption is one such popular technique that can be used to provide a guarantee that data will not be lost to intruders. AES algorithm is ciphered with a structure that is given, and the maximum security is offered to data. In order to accomplish that it is founded on the rounds which are composed of four sub-processes. Each round contains the following four steps towards encrypting a 128-bit block.

A. Substitute Bytes Transformation

The first stage is the SubBytes transformation that is used in each round. This stage is based on the nonlinear S-box which substitutes one of the bytes of the state with another. The diffusion and confusion indicate that the design principles of the cryptographic algorithms of Shannon are important to attain much higher degree of security. As an example, in AES, when the hex 53 appears in the state it should

be replaced by hex ED. The intersection point of 5 and 3 is an ED. These operations should be carried out on the rest of the bytes of the state.

B. Shift Rows Transformation

Next operation to be applied to the state is the operation that is applied after SubByte and is called ShiftRow. The reason is that state bytes are rotated in a cyclic manner to the left in each row, beginning with row 0. The permutation is not on the bytes located in the row 0 and the lost bytes are not lost. In a line only one byte is transmitted to the left in the first row. The second row has been moved by two-byte to the left. Three bytes are shifted leftwards pushing the third row. The state size does not change, i.e. the state is of an original length of 16 bytes only that the location of bytes within the state is repositioned.

C. Mix Columns Transformation

The other significant step of the state is the Mix Column step. Calculation of multiplication takes place in the state. Each byte of one row of the matrix transformation multiplies the value of individual bytes of the state column. This implies that each row of the mean transformation has to be multiplied by all the columns. All these multiplications are XORed in order to create a 4-byte value of the next state. In this step, the state size is maintained to be 4x4.

D. Add Round Key Transformation

Add Round Key is the most significant AES algorithm step. The key and the input data (also called the state) is presented in the form of a 4x4 array of bytes. The main and input data (128 bits) is divided into a matrix of bytes. Add Round Key is much more robust in terms of providing data encryption. The operation will create a correlation between the key and the ciphertext. The former step is delivery of the ciphertext. The results of the Add Round Key are totally dependent on the key that the user typed in. The subkey will also be utilized in this stage and will be changed along with the state. Every round is constructed based on the subkey of the last round with the help of the Rijndael key schedule. The size and the state of the subkey are the same. XORing the state and subkey bytes is added to the subkey to make the subkey.

3.4.2. AES Key Expansion

The encryption and the decryption of AES algorithm are founded on the expansion of key. This is the other significant step of AES structure. Each round has a new key. This section is concerned with AES key expansion method. The massive expansion process generates word-based round keys and each word is a set of 4-bytes. The repetition generates $4x(Nr + 1)$ words. Where Nr is the rounds. It works in the following way: The first four words are obtained with the help of the cipher key (initial key). This key consists of 16 bytes ($k0 - k15$) that is revealed in figure 8 and presented in the form of an array. The first four bytes ($k0$ to $k3$) are called the w0, the next four bytes ($k4$ to $k7$) correspond to the first column and so on. The keys of all the rounds are not difficult to compute and to know, and the following formula may be used in equation (19):

$$K[n]: w[i] = k[n-1]: w[i] XOR k[n]: w[i] \quad (19)$$

3.4.3. Decryption Process

Decryption is a process that entails the extraction of the original information in an encrypted file. This is done through the assistance of the key of the data sender. The AES decryption process is the opposite

of encryption process and both the sender and the receiver are using the same key in encryption and decryption of data. The third and the last step of the decryption section comprises of three steps, which are InvShiftRows, InvSubBytes and AddRoundKey, as depicted in figure 3.

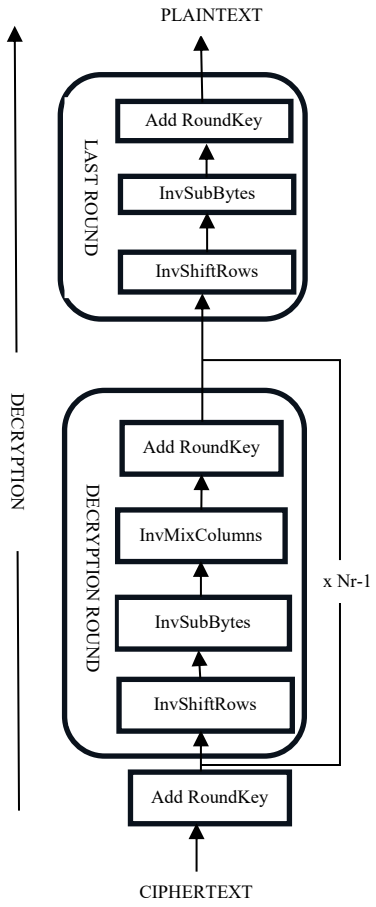

Figure 3: Decryption process

3.4.4. Improved Advanced Encryption Standard (IAES)

Design the Better AES Algorithms the AES encryption is a famous symmetric-key encryption. The most crucial arrangement of the AES encryption algorithm is presented in figure 4. To achieve a uniform distribution of ciphertext, a sequence of intertwined operations was added in the creation of AES cryptosystem. Part of the operations that were related to this were the replacement of the inputs with an insignificant amount of output feedback (substitutions). The left operations had to do with bit permutation. AES computes using the use of bytes. As such, AES was dividing a block of information of 128 bits into 16 bytes. These 16 bytes was then preceded by a 4-by-4 matrix of the operation. AES cryptosystem was comprised of several rounds, depending on key length within which were dealing. Figure 4 reveals that the key length of 10, 12 and 14 rounds was 128, 192 and 256 bits respectively. The execution of every round was done using other round keys, KR, which was the derivation of the starting fixed key of AES, and R was the round number.

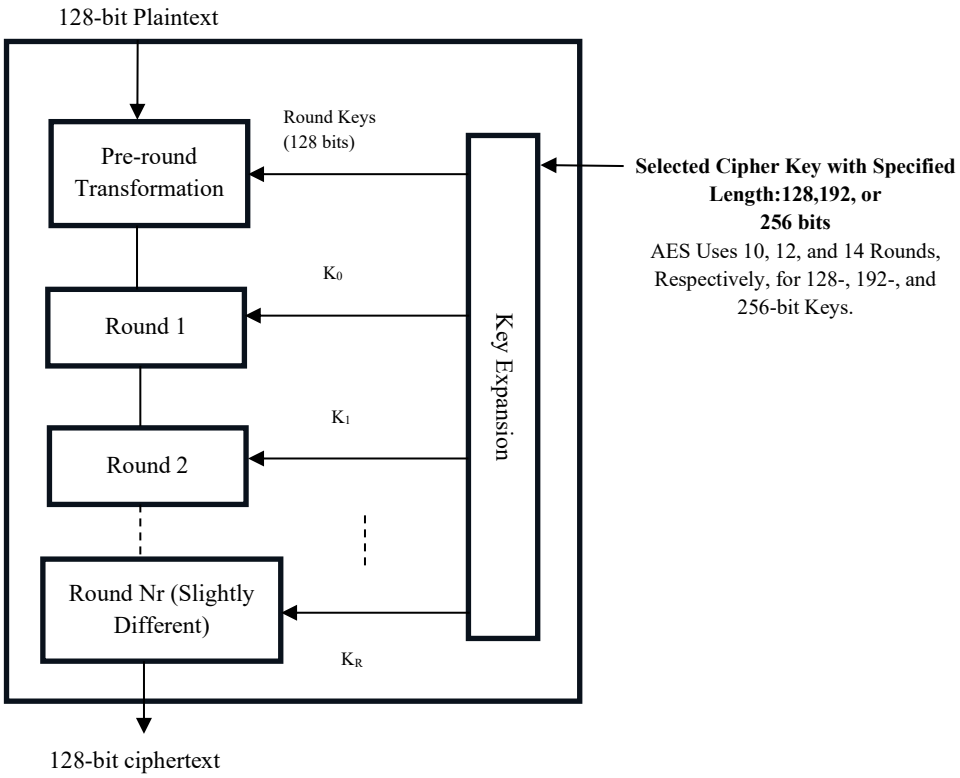

Figure 4: Advanced encryption standard (AES) architecture

AES cryptosystems with symmetric key used the keys as common secrets within the communication system that secured the information being communicated. But in contrast to the metric-key encryption, the cryptosystems with a symmetric key use a secret key. All the information will be divulged once this predetermined secret key is stolen. So, a new IAES encryption algorithm is recommended to resolve this issue and synchronization chaotic dynamic keys are used in this paper as demonstrated in figure 5.

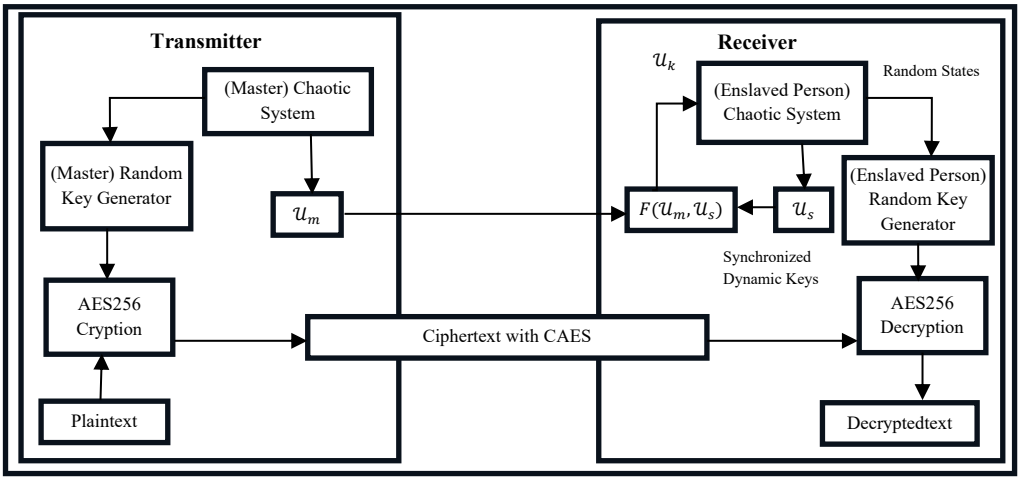

Figure 5: Improved commotion-based AES architecture

In the suggested improved commotion-based AES architecture the dynamic random states of synchronized chaos systems were the secure keys. Evidently, the enhanced design has the potential of supplying synchronized random states to a random key generator that will in effect generate dynamic

256-bit AES key to enhance safety. The achievement of a master-slave system could be done by matching the slave states to the master states using a designed controller within the slave communication system. To enhance the security of practical applications, the synchronization controller ($u(k)$) was decomposed into $u_m(k)$ and $u_s(k)$, which are derived from the enslaver and enslaved person commutation systems, respectively. The receiver-end controller implementation was then adopted: $u(k) = F(u_m(k), u_s(k))$. The enslaver and enslaved person system states were not separable to hackers and even when such parameters could be present, were not reconfigurable by the synchronization controller. By so doing, this would guarantee the encryption system and protection of the dynamic keys. Under synchronizing, synchronous reception of the identical random commotion signals are realizable and the dynamic random keys of the transmitter and receiver may be synchronized. The predefined keys of the former AES were replaced by the randomly coordinated signals that were not obliged to be provided in advance and sent on the open events. Thus, the drawback of key storage may be averted, and the security of encryption with dynamically-generated keys may be raised with the help of synchronization. AES algorithm invention of commotion is more secure when compared to the current processes.

3.5 Pseudocode of Proposed EDFO Algorithm

Algorithm: Enhanced Dragonfly–Firefly Optimization (EDFO)

Input: Set of tasks T, set of virtual machines VM

Output: Optimal task scheduling with balanced load

1. Initialize population of dragonflies (solutions)

2. Assign tasks randomly to VMs

3. Evaluate fitness function (makespan, load balance, cost)

4. While (termination criteria not met) do

a. For each solution:

i. Compute separation, alignment, cohesion

ii. Compute attraction to food source

iii. Compute the distraction from the enemy

iv. Update velocity and position (Dragonfly update)

b. Apply Firefly optimization:

i. Compare the brightness (fitness) of solutions

ii. Move weaker solutions toward stronger ones

iii. Apply randomization for exploration

c. Recalculate fitness for updated solutions

d. Select the best solution:

If (FA fitness < DA fitness)

Replace the DA solution with the FA

Else

Retain the DA solution

5. End While

6. Return the best task-to-VM mapping

The EDFO algorithm combines the Dragonfly exploration algorithm with the Firefly exploitation algorithm. Dragonfly elements enable global search, and Firefly enhances convergence on the best solutions. The hybrid mechanism increases load balancing, decreases makespan, and maximizes resource utilization.

3.6 Parameter Initialization and Control Settings

To stabilize and ensure optimal operation of the proposed EDFO algorithm, the most important control parameters are initialized using empirical tuning and previous results in swarm intelligence optimization. The choice of parameters determines the speed of convergence, the exploration-exploitation trade-off, and the quality of solutions.

Table 2: Parameter initialization of the EDFO algorithm

Parameter	Value
Population Size	30
Maximum Iterations	100
Inertia Weight	$0.9 \rightarrow 0.4$
Attraction Coefficient	1.0
Randomization Parameter	0.2

In table 2 indicates that the population size is chosen to balance computational cost and solution diversity. The inertia weight is reduced linearly to improve convergence. The attraction coefficient and randomization parameter control Firefly behavior, enabling it to search the entire world without prematurely converging on a single point.

4 Results and Discussion

To explore the applicability of the proposed algorithm to a real-life setting, the experiments were carried out using the CloudSim-3.0.3 simulator (Chafi et al., 2021), which is tested and proven by several other researchers. In addition, the experimental data were generated randomly to test the performance of the proposed algorithm in the reality world. The. The various traces of workloads on a real world were experimented using the CloudSim-3.0.3 simulator (Chafi et al., 2021) that has been utilized by various researchers. Also, randomly generated original data have been used in the experiments in CloudSim. The parameters that could be represented by the numbers in braces are as follows: the first number relates to experiments when the independent task is manipulated, and the impact of manipulating the coefficient is evaluated.

The second value, on the contrary, is associated with the experiment of the variable workload and different numbers of VMs. The former is the first value that is taken into account in terms of the first configuration and the balance of the distribution of VMs in specific hosts. The second value corresponds to the second configuration and imbalance of VMs distribution across hosts.

CloudSim (version 3.0.3) was used as a simulation framework in Java (JDK 1.8) within the Eclipse IDE (2021-06). The experiments were run on a system powered by an Intel Core i7 processor, 8 GB of RAM, and Windows 10. This system is designed to provide a stable test of the offered EDFO-IAES algorithm in real conditions of a cloud computing system.

In table 3 compares the imbalance level of the proposed EDFO-IAES approach with state-of-the-art approaches such as DFGA (Marri & Rajalakshmi, 2022) and IBPSO-LBS (Menaka & Kumar, 2024). Believe that the better result of EDFO-IAES in table 3 can be explained by the hybrid EDFO mechanism,

where local exploitation is fostered by Firefly optimization, and the global exploration is provided by Dragonfly optimization. This combination successfully allocates the workload between the VMs, thereby reducing imbalance compared to DFGA and IBPSO-LBS.

Table 3: Comparison of the degree of imbalance with existing methods DFGA (Marri & rajalakshmi, 2022) and IBPSO-LBS (Menaka & kumar, 2024)

No. of. Workloads	DFGA	IBPSO-LBS	EDFO-IAES
W1	3.9	2.7	1.8
W2	3.3	2.2	1.4
W3	3.5	2.5	1.2
W4	3.1	2.6	1.5

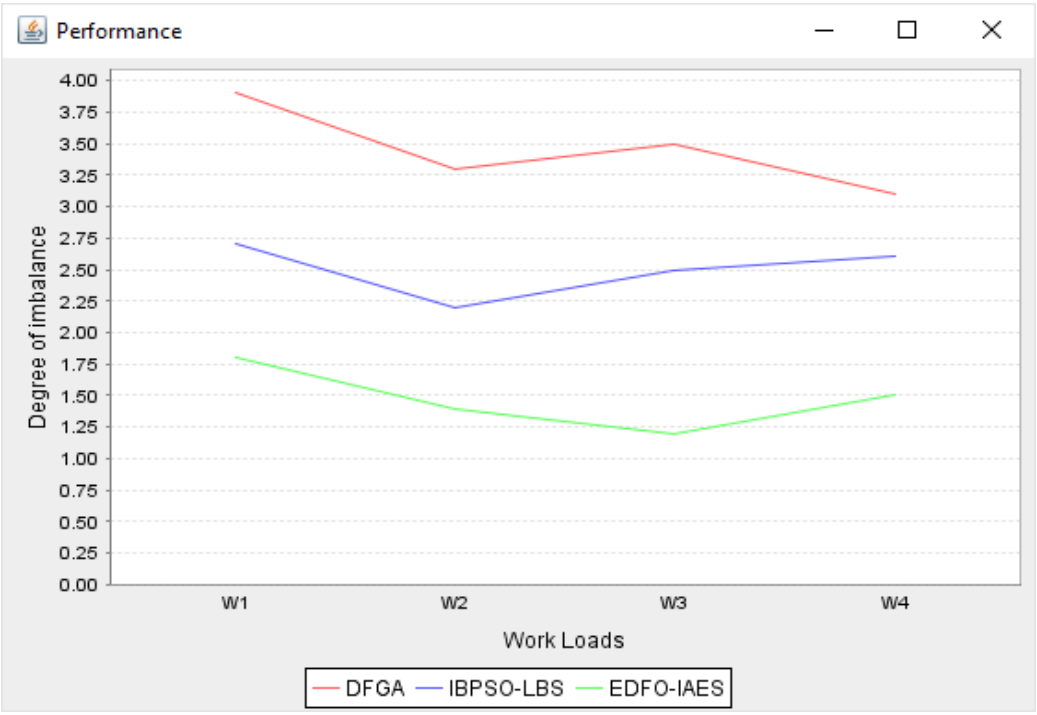

Figure 6: Degree of imbalance comparison results between the proposed and existing methods

In figure 6 reveals that the suggested EDFO-IAES algorithm minimizes imbalance (DI) as compared to DFGA and IBPSO-LBS. In this experiment, the number of loads is growing resulting in increased incoming user tasks with limited capacity. Thus, the proposed EDFO-IAES algorithm is to be monitored in this experiment to identify the improvement in its performance with the growth in the number of tasks. In the meantime, the hosts are almost equal in their capacity as indicated by the number of VMs and the total processing speed of the VMs host. With this experiment, the slight improvement was achieved in the proposed algorithm on EDFO. This is due to the fact that the EDFO is not only allocating the same hosts but the overall length of the user tasks as well. The suggested EDFO-IAES has a major betterment over the other existing algorithms.

In table 4 compares the average storage utilization of the proposed EDFO-IAES method with state-of-the-art methods DFGA (Marri & Rajalakshmi, 2022) and IBPSO-LBS (Menaka & Kumar, 2024). The increased storage utilization achieved by EDFO-IAES is attributed to its task-to-VM

mapping strategy, which is dynamically evaluated based on load. The objective function aims for a balanced allocation and results in improved resource use compared to current methods.

Table 4: Comparison of average storage utilization with existing methods DFGA (Marri & rajalakshmi, 2022) and IBPSO-LBS (Menaka & kumar, 2024)

No. of. workloads	DFGA	IBPSO-LBS	EDFO-IAES
W1	82	89	94
W2	84	85	92
W3	83	84	93
W4	81	86	91

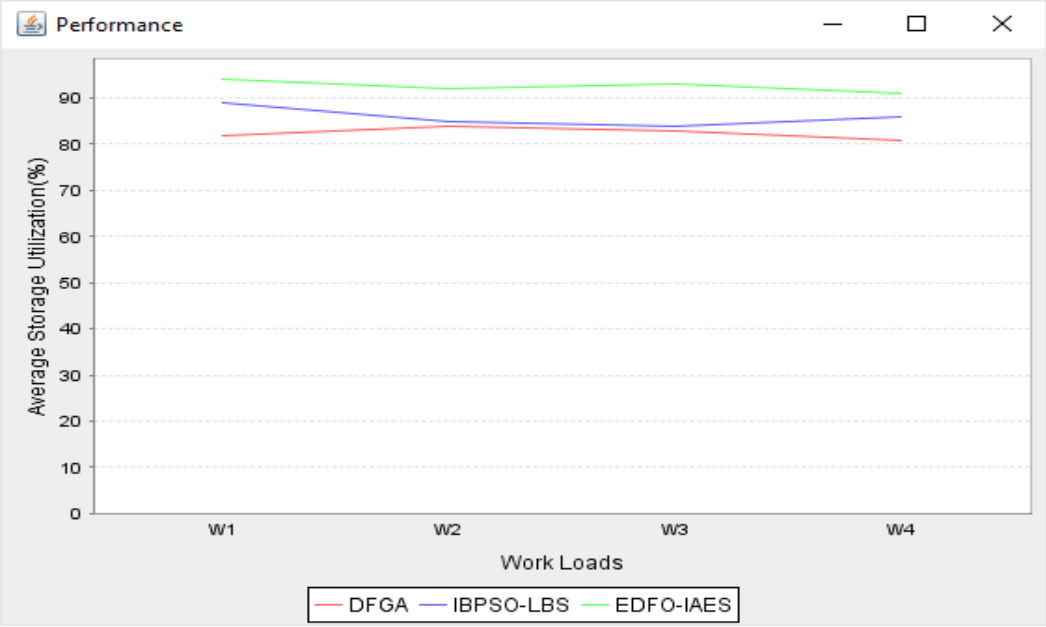

Figure 7: Average storage utilization comparison results between the proposed and existing methods

The average storage utilization rate of EDFO-IAES is significantly more than average storage utilization rate of other ones, as shown in figure 7. The nearest scores are 100% in EDFO-IAES. This means that the cloud platform does not go to waste when the other workloads are executed and a high rate of utilization is achieved. The EDFO-IAES algorithm is, thus, capable of optimizing the use of the available heterogeneous resources. This is a plus to providers: it ensures that resources are employed and allows them to earn as much profit as can by outsourcing a limited number of resources.

Table 5: Comparison of makespan with existing methods DFGA (Marri & rajalakshmi, 2022) and IBPSO-LBS (Menaka & kumar, 2024)

No. of. tasks	DFGA	IBPSO-LBS	EDFO-IAES
1000	224	198	157
2000	245	212	176
3000	273	235	188
4000	296	256	205

In table 5 compares the makespan of the proposed EDFO-IAES technique with state-of-the-art methods, including DFGA (Marri & Rajalakshmi, 2022) and IBPSO-LBS (Menaka & Kumar, 2024). The decrease in the makespan (Table 5) is directly due to the multi-objective optimization of the EDFO

algorithm, which minimizes the time spent executing tasks without violating load balance. The hybrid search approach can accelerate the process of arriving at optimal scheduling decisions.

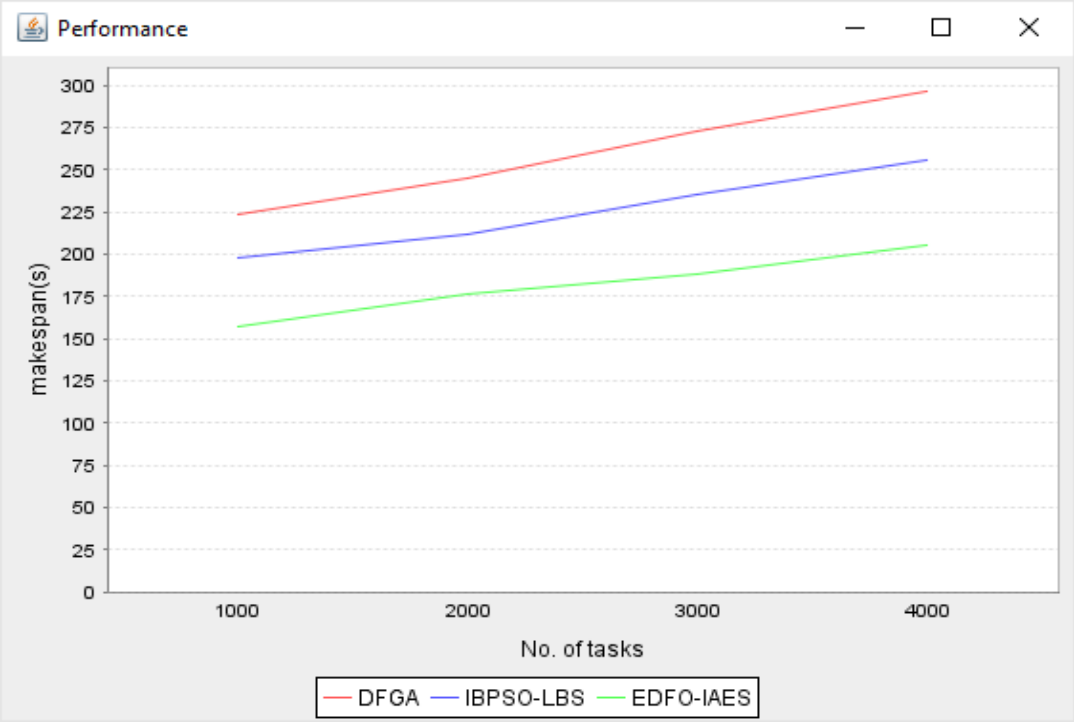

Figure 8: Makespan comparison results between the proposed and existing methods

In figure 8 indicates that makespan is reduced with the number of tasks in the system because the larger processing power can complete the number of tasks assigned to it. Based on the results, the proposed EDFO-IAES algorithm is capable of achieving the minimal makespan and being more efficient than IBPSO-LBS and DFGA. To test the scalability of the proposed EDFO-IAES algorithm, the experiment also multiplied the number of virtual machines (VMs into the system) to add processing power and, therefore, faster task processing and more tasks.

Table 6: Comparison result of average waiting time (s)

No. of. VMs	DFGA	IBPSO-LBS	EDFO-IAES
25	1825	1489	1326
50	1756	1365	1139
75	1621	1286	963
100	1482	1214	897

In table 6 tabulates the comparison results for average waiting times between the proposed and existing methods.

In figure 9 indicates that the mean waiting time of the EDFO-IAES algorithm is lower than the other algorithms. As have found in the provided experiment and analysis, the created EDFO-IAES algorithm can also be applied to the management of the increased resource demand, which also includes the virtual machines. In this way, it is possible to conclude that the scalability can be enhanced with the assistance of the suggested algorithm. Sum up all the results share in the fact that the suggested IBPSO-LBS algorithm can be productive in maintaining the load balance under the real workloads, random data sets,

and the heterogeneous resources- the predictors of the high performance of the proposed algorithm, EDFO-IAES.

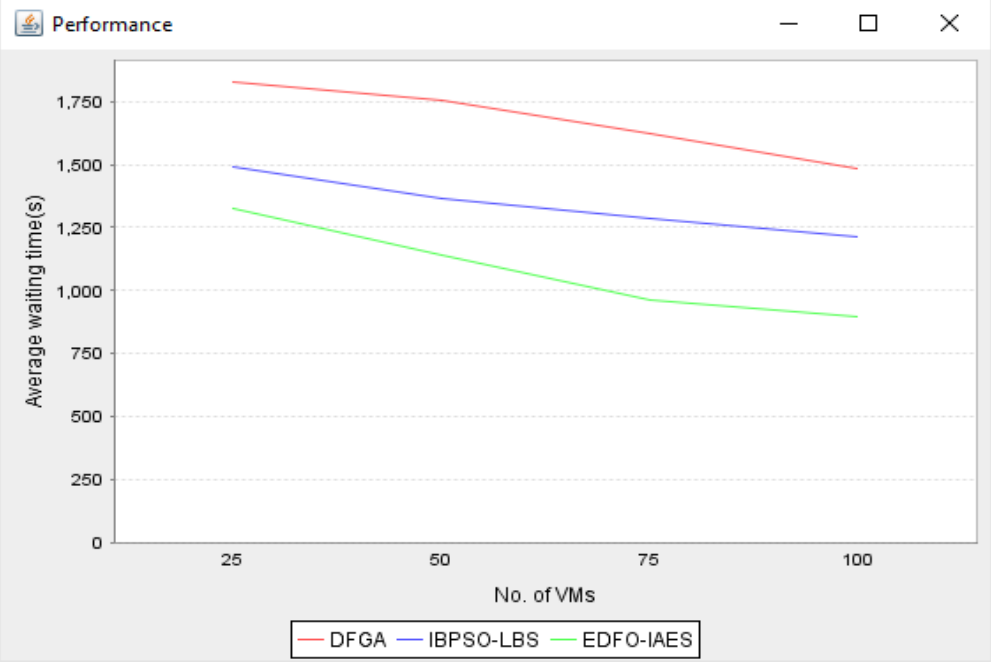

Figure 9: Average waiting time comparison results between the proposed and existing methods

Ablation Study

An ablation study is conducted to assess the role of individual mechanisms in the proposed EDFO-IAES model by selectively turning off key mechanisms.

1. In the absence of Firefly Optimization (EDGO → DO only): The Firefly part of the Firefly component decreases the local search, resulting in a slower convergence and a higher makespan, which should emphasize the essentiality of exploitation in polishing solutions.
2. In the absence of Dragonfly Optimization (EDFO - FO = FO), global exploration is capped by the removal of the Dragonfly aspect, leading to a suboptimal task split and greater imbalance, thereby proving that successful exploration requires optimization.
3. In the absence of an IAES Strategy (EDFO without adaptive scheduling), resource utilization is less effective, and execution time is longer, corroborating the role of IAES-based adaptive scheduling in dynamic load balancing and decision optimization.

5 Discussion

As the experimental results show, the proposed EDFO-IAES model is much more efficient than the existing models, namely DFGA and IBPSO-LBS, across vital performance indicators such as imbalance degree, storage utilization, and makespan. This is primarily due to the hybrid optimization method, which effectively balances global exploration (Dragonfly Optimization) and local exploitation (Firefly Optimization) to achieve optimal task scheduling decisions.

The task division in the methodological approach to integrating IAES also greatly improves flexibility, as the distribution is dynamically recalculated based on the system's state. This will result in

efficient resource utilization and reduced execution time, thereby improving the efficiency of the proposed multi-objective optimization framework.

Several challenges remain. The algorithm contains several control parameters that can be effectively tuned, which may differ depending on the environment. Another possibility is that large workloads can increase the computational overhead of hybrid optimization. The model is being tested in a simulated environment, and its application to the real world may introduce uncertainties, including network latency, node failures, and dynamically changing workloads.

The generalizability of the findings is limited in some ways. This experiment fails to consider energy efficiency, fault tolerance, or security, which are relevant in a real-world cloud system. In the future, this should be enhanced by generalizing the model to real-time cloud environments, parameter tuning that is adaptive, and other quality of service constraints that can be used to ensure the model is robust and scalable.

Limitations of the Study

Although the proposed EDFO-IAES model has high potential, several constraints may affect its external validity. To start with, the experiment is conducted in a simulated cloud environment that may not be sufficiently representative of the nature and dynamic uncertainties of real-life cloud infrastructures. Second, the model is tested only on a narrow set of workloads, and its performance can change when applied to a highly heterogeneous or even a large distributed system. Third, the EDFO algorithm employs empirical tuning as a parameter setting, which may affect performance across various scenarios. Furthermore, the current model does not specifically address network latency, energy consumption, or fault tolerance. The additional research will be guided towards real-time implementation, parameter optimization, and the addition of other quality of service indicators to render the system strong and practical.

Practical Relevance and Deployment Feasibility

The proposed EDFO-IAES model is highly practical and can be implemented in cloud systems, such as large-scale data centers, distributed computing infrastructures, and edge-cloud systems, as it optimizes task scheduling, resource utilization, and resource span. The algorithm can be used with the current cloud schedulers with minimal alterations, as it operates with standard workload and resource parameters. Its elasticity with IAES would enable it to dynamically respond to workload changes, which is essential for real-time applications such as IoT, big data analytics, and e-governance systems. Although the computational cost is high, it is not prohibitive in modern cloud infrastructures, which is why the model can be scaled and offers practical advantages.

6 Conclusion

To address the task scheduling, load balancing, and security challenges in a cloud computing environment, this paper proposes an Enhanced Dragonfly-Firefly Optimization with Improved Advanced Encryption Standard (EDGO-IAES). The results of the experiment show that the model is highly effective at improving the system's performance compared to existing approaches such as DFGA and IBPSO-LBS. Specifically, the imbalance is reduced by 40-45%, the average storage utilization is boosted to 94%, and the makespan is reduced by 20-25% across different workloads. The results of the above enhancements indicate that the hybrid EDFO algorithm can achieve optimal task allocation and effective resource utilization. The IAES combination guarantees the safety and integrity of data

transmission without placing excessive computational load, thereby improving the overall reliability of the system. The proposed framework also offers reduced execution time, reduced cost-complexity, and enhanced throughput; therefore, it is suitable for a large, dynamic cloud environment. The EDFO-IAES model is a highly efficient, effective, and safe way to schedule and load-balance tasks in clouds. Future studies will aim to generalize the model to real-time cloud environments, optimize model parameters, and apply sophisticated hybrid optimization and encryption techniques to address source allocation, energy efficiency, and network-level concerns, e.g., packet.

References

[1] Adee, R., & Mouratidis, H. (2022). A dynamic four-step data security model for data in cloud computing based on cryptography and steganography. *Sensors*, *22*(3), 1109. https://doi.org/10.3390/s22031109

[2] Akilashri, P. S. S., & Ahadha Parveen, A. (2022, August). A study of Different Encryption Techniques to Generate Keys in Cloud Computing Security. In *Proceedings of International Conference on Computational Intelligence, Data Science and Cloud Computing: IEM-ICDC 2021* (pp. 387-397). Singapore: Springer Nature Singapore. https://doi.org/10.1007/978-981-19-1657-1_34

[3] Alkhatib, A. A., Alsabbagh, A., Maraqa, R., & Alzubi, S. (2021). Load balancing techniques in cloud computing: Extensive review. *Advances in science, technology and engineering systems journal*, *6*(2), 860-870. https://dx.doi.org/10.25046/aj060299

[4] Chafi, S. E., Balboul, Y., Mazer, S., Fattah, M., Bekkali, M. E., & Bernoussi, B. (2021). Cloud computing services, models and simulation tools. *International Journal of Cloud Computing*, *10*(5-6), 533-547. https://doi.org/10.1504/IJCC.2021.120392

[5] Devaraj, A. F. S., Elhoseny, M., Dhanasekaran, S., Lydia, E. L., & Shankar, K. (2020). Hybridization of firefly and improved multi-objective particle swarm optimization algorithm for energy efficient load balancing in cloud computing environments. *Journal of Parallel and Distributed Computing*, *142*, 36-45. https://doi.org/10.1016/j.jpdc.2020.03.022

[6] Farajizadeh, M., & Bakhsh, N. N. (2015). A mechanism to improve the throughput of cloud computing environments using congestion control. *International Academic Journal of Science and Engineering*, *2*(1), 97-111.

[7] Ghaseminya, M. M., Eslami, E., Shahzadeh Fazeli, S. A., Abouei, J., Abbasi, E., & Karbassi, S. M. (2025). Advancing cloud virtualization: a comprehensive survey on integrating IoT, Edge, and Fog computing with FaaS for heterogeneous smart environments. *The Journal of Supercomputing*, *81*(14), 1303. https://doi.org/10.1007/s11227-025-07799-2

[8] Hammouri, A. I., Mafarja, M., Al-Betar, M. A., Awadallah, M. A., & Abu-Doush, I. (2020). An improved dragonfly algorithm for feature selection. *Knowledge-based systems*, *203*, 106131. https://doi.org/10.1016/j.knosys.2020.106131

[9] Karunya, N., & Devi, D. G. (2025, September). F-FLAME: Intelligent Fuzzy-Hybrid Optimization for Sustainable VM Migration in Federated Cloud Systems. In *2025 International Conference on Transformative Computing Technologies (ICTCT)* (pp. 315-322). IEEE. https://doi.org/10.1109/ICTCT69201.2025.00066

[10] Kondori, M. A., & Peashdad, O. H. (2015). Analysis of challenges and solutions in cloud computing security. *International Academic Journal of Innovative Research*, *2*(1), 20-30.

[11] Mandal, G., Dam, S., Dasgupta, K., & Dutta, P. (2023, January). Load Balancing in a Heterogeneous Cloud Environment with a New Cloudlet Scheduling Strategy. In *International Conference on Computational Intelligence in Communications and Business Analytics* (pp. 102-117). Cham: Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-48879-5_9

[12] Manoj Kumar, T., & Karthigaikumar, P. (2020). A novel method of improvement in advanced encryption standard algorithm with dynamic shift rows, sub byte and mixcolumn operations for

the secure communication. *International Journal of Information Technology*, *12*(3), 825-830. https://doi.org/10.1007/s41870-020-00465-1

[13] Marri, N. P., & Rajalakshmi, N. R. (2022). MOEAGAC: an energy aware model with genetic algorithm for efficient scheduling in cloud computing. *International Journal of Intelligent Computing and Cybernetics*, *15*(2), 318-329. https://doi.org/10.1108/IJICC-07-2021-0134

[14] Masadeh, E., Masadeh, R., Almomani, O., Alshqurat, K., & Masadeh, S. (2025). Cloud Computing Task Scheduling using Genetic Algorithm: A Survey. *Journal of Humanities and Social Sciences Studies*, *7*(6), 74-87. https://doi.org/10.32996/jhsss.2025.7.6.8

[15] Menaka, M., & Kumar, K. S. (2024). Supportive particle swarm optimization with time-conscious scheduling (SPSO-TCS) algorithm in cloud computing for optimized load balancing. *International Journal of Cognitive Computing in Engineering*, *5*, 192-198. https://doi.org/10.1016/j.ijcce.2024.05.002

[16] Mishra, S. K., GC, V. J. D., Maddi, P. A., Tanniru, N. M., & Manthena, S. L. P. (2024, January). Enhancing edge intelligence with layer-wise adaptive precision and randomized PCA. In *2024 International Conference on Advancements in Smart, Secure and Intelligent Computing (ASSIC)* (pp. 1-5). IEEE. https://doi.org/10.1109/ASSIC60049.2024.10507942

[17] Mishra, S. K., Sahoo, B., & Parida, P. P. (2020). Load balancing in cloud computing: a big picture. *Journal of King Saud University Computer and Information Sciences*, *32*(2), 149-158. https://doi.org/10.1016/j.jksuci.2018.01.003

[18] Nehra, P., & Kesswani, N. (2023). Efficient resource allocation and management by using load balanced multi-dimensional bin packing heuristic in cloud data centers. *The Journal of Supercomputing*, *79*(2), 1398-1425. https://doi.org/10.1007/s11227-022-04707-w

[19] Nissar, S., Ahmed, B., Sood, S., Kaur, B., & Brar, M. K. (2024, July). Security techniques in cloud computing. In *AIP Conference Proceedings* (Vol. 3121, No. 1, p. 030009). AIP Publishing LLC. https://doi.org/10.1063/5.0221528

[20] Pattnaik, P. K., Le, D. N., & Pal, S. (2022). Security paradigms in cloud computing. *Cloud Computing Solutions: Architecture, Data Storage, Implementation and Security*, 181-196. https://doi.org/10.1002/9781119682318.ch11

[21] Saboor, A., Mahmood, A. K., Omar, A. H., Hassan, M. F., Shah, S. N. M., & Ahmadian, A. (2022). Enabling rank-based distribution of microservices among containers for green cloud computing environment. *Peer-to-Peer Networking and Applications*, *15*(1), 77-91. https://doi.org/10.1007/s12083-021-01218-y

[22] Shah, S., Das, P., Latoria, A., & Thaker, D. J. (2024). Optimized load balancing techniques in cloud computing environment: A systematic literature review and future trends. *Towards Excellence*, *16*(3). https://doi.org/10.37867/TE160312

[23] Sharma, T., & Bedi, R. S. (2024). Design and development of pragmatic load balancing algorithm for cloud environment. *Wireless Personal Communications*, *136*(1), 81-101. https://doi.org/10.1007/s11277-024-11117-z

[24] Singamaneni, K. K., Muhammad, G., & Ali, Z. (2024). A novel quantum hash-based attribute-based encryption approach for secure data integrity and access control in mobile edge computing-enabled customer behavior analysis. *IEEE Access*, *12*, 37378-37397. https://doi.org/10.1109/ACCESS.2024.3373648

[25] Tian, Y., & Nogales, A. F. R. (2023, March). A Survey on Data Integrity Attacks and DDoS Attacks in Cloud Computing. In *2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC)* (pp. 0788-0794). IEEE. https://doi.org/10.1109/CCWC57344.2023.10099240

[26] Wang, Z., Chen, S., Bai, L., Gao, J., Tao, J., Bond, R. R., & Mulvenna, M. D. (2023). Reinforcement learning based task scheduling for environmentally sustainable federated cloud computing. *Journal of Cloud Computing*, *12*(1), 174. https://doi.org/10.1186/s13677-023-00553-0

[27] Yang, X. S., & Slowik, A. (2020). Firefly algorithm. In *Swarm intelligence algorithms* (pp. 163-174). CRC Press.

Authors Biography

P. Viswanatha Reddy, Research Scholar in the Department of Networking & Communications in the School of Computing, SRM Institute of Science and Technology, Kattankulathur-603203, Tamil Nadu, INDIA. He received a B.Tech. degree from Sri Sai Institute of Technology & Science, Rayachoty in 2008. He was awarded an M.Tech from Bharath University, Chennai, in 2011. He is pursuing Ph.D at SRM Institute of Science and Technology. He has 15 Years of Teaching experience. His area of Interest lies in cloud computing, and published 15 papers in national journals and presented 6 papers in national and international conferences in that area.

Dr.P. Savaridassan is currently working as an Assistant Professor at SRM Institute of Science and Technology, where he has been serving since 2013. He has over a decade of academic and research experience in the field of computer science and information technology. His research interests include digital forensics, cybersecurity, cloud computing, data mining, and machine learning. He has authored and co-authored several research papers in reputed journals and conferences, focusing on areas such as anomaly detection, intelligent systems, and secure computing frameworks. Dr. Savaridassan is actively involved in mentoring undergraduate and postgraduate students and has contributed to curriculum development and academic initiatives. His current research focuses on developing innovative and scalable solutions for cybersecurity and data-driven applications.