National College of Ireland

Optimizing Load Balancing in cloud computing using Enhanced Firefly Algorithm (EFA) Method

MSc Research Project
Cloud Computing

Jayant Shah
Student ID: x21219583

School of Computing
National College of Ireland

Supervisor: Prof Aqeel Kazmi

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Jayant Shah
Student ID:	x21219583
Programme:	Cloud Computing
Year:	2023
Module:	MSc Research Project
Supervisor:	Prof Aqeel Kazmi
Submission Due Date:	14/12/2023
Project Title:	Optimizing Load Balancing in cloud computing using Enhanced Firefly Algorithm (EFA) Method
Word Count:	5995
Page Count:	21

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	13th December 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Optimizing Load Balancing in cloud computing using Enhanced Firefly Algorithm (EFA) Method

Jayant Shah
x21219583

Abstract

Cloud computing has seen a surge in interest from both the academic and business sectors due to its scalability and virtualization benefits. Efficient load balancing within cloud data centers is essential to maintain system stability and performance. Load Balancing (LB) is particularly critical in cloud environments where providers serve a multitude of clients, necessitating robust task scheduling that ensures equitable load distribution. Various strategies and algorithms have been developed to advance LB in cloud services, focusing on minimizing task execution time, reducing energy consumption, optimizing resource use, and rapidly allocating tasks among clusters of Virtual Machines (VMs). Nonetheless, these solutions often overlook the existing load on VMs, potentially leading to overload issues. To address this, the research introduces the Enhanced Firefly Algorithm (EFA), a refined approach leveraging Metaheuristic Optimization. EFA intelligently evaluates VM workloads in cloud infrastructures to distribute tasks without overloading any single VM. It is compared against traditional LB algorithms like Round Robin Algorithm (RRA) and Ant Colony Algorithm (ACO), using metrics such as resource consumption, computational speed, and time efficiency. The results demonstrate EFA's superior performance over RRA and ACO, particularly as the number of task units (cloudlets) increases. EFA achieves better execution times, enhanced resource utilization, and more efficient LB solutions. This study underscores the necessity for LB mechanisms that consider the current load of VMs alongside resource, timing, and energy metrics, presenting EFA as a viable solution for dynamic and efficient load distribution in cloud computing.

1 Introduction

Cloud Computing (CC) has fundamentally altered digital interaction, making computing accessible like a utility service and enabling scalable, on-demand resources over the Internet, which is pivotal in driving digital transformation across various sectors. As CC continues to evolve, it is increasingly integrating into the core of modern computing services. One of the critical challenges emerging with this expansion is Load Balancing (LB). LB plays a vital role in ensuring fair workload distribution across Virtual Machines (VMs) in cloud environments. Effective LB dynamically allocates tasks, maintaining system agility and stability by adapting to fluctuating computing demands. However, achieving this dynamic equilibrium is complex, especially with unpredictable task demands that risk overloading some VMs while under utilizing others. Traditional LB strategies often focus merely on task allocation, neglecting the crucial balance across the

VM cluster, leading to inefficient resource use. Consequently, users may experience increased wait times and higher operational costs. Therefore, there's a pressing need for LB mechanisms that proactively manage ongoing loads within the VM cluster, not just react to new tasks. Such holistic LB approaches can preempt bottlenecks and optimize resource distribution, enhancing overall system performance and user experience.

As CC continues to integrate into the core of service delivery, the need for advanced LB techniques becomes ever more pressing. Businesses and service providers must embrace strategies that can navigate the complexities of a distributed computing landscape. This includes leveraging sophisticated algorithms that can predict and adapt to changing load patterns, thereby maintaining an equilibrium that ensures efficiency and reduces latency. Moreover, the integration of artificial intelligence and machine learning into LB processes holds promise. These technologies could offer adaptive mechanisms that not only respond to immediate load changes but also learn and anticipate future demands. Such predictive capabilities would mark a paradigm shift in how Load Balancing is approached, transforming it from a challenge to a strategic asset in cloud computing.

As CC matures and expands, it must be accompanied by LB solutions that are as dynamic and flexible as the services they support. Future-focused LB strategies will play a pivotal role in ensuring that cloud environments remain robust, efficient, and capable of meeting the evolving demands of users. This will be instrumental in harnessing the full potential of cloud computing, enabling it to continue its trajectory as a cornerstone of digital infrastructure.

1.1 Background

Various approach to task scheduling in cloud computing were developed by integrating an Osmotic Bio-inspired algorithm by Ojha et al. (2020) .The approach was refined by enhancing the Osmotic Algorithm with the Firefly algorithm.The study delves into existing load balancing techniques, examining their strengths and weaknesses. Although the suggested method accomplished its primary goal but the drawback was that at higher workloads, the system risked getting stuck in local optima, potentially compromising the overall effectiveness of the load balancing. Alameen and Gupta (2020)study, introduced an improved meta-heuristic algorithm known as the Fitness Rate-Based Rider Optimization Algorithm (FR-ROA), an iteration of the Rider Optimization Algorithm (ROA). They assessed its efficiency by measuring the time to complete individual tasks as well as the total time to complete all tasks. Their research indicated that the FR-ROA performs effectively with small-scale tasks and within short timeframes. However, the algorithm was found to be complex, leading to issues related to computational complexity. Perepelkin and Nguyen (2022), has developed and analyzed a multipath routing and load balancing algorithm for Software Defined Networks (SDN) using an Artificial Bee Colony (ABC) algorithm, inspired by the foraging behavior of honey bees. The ABC model incorporates different bee roles—employed, onlooker, and scout bees—to optimize routing paths. They utilized priority coding for path selection and conducted simulations to demonstrate the effectiveness of their algorithm in reducing route congestion and increasing data packet transmission across multiple routes. The main problem the research addressed was optimizing multipath routing Perepelkin et al. (2023)to prevent route congestion and improve overall network traffic management in SDNs. The research highlighted certain shortcomings, such as substantial expenses, inefficient processing durations, and a lack of attention to the current burden on Virtual Machines, which can

lead to overload problems. These issues include the financial burden, suboptimal time management in task execution, and the oversight of pre-existing workloads on the VMs that could result in overloading.

1.2 Motivation

To optimize resource use and balance workloads across Virtual Machines (VMs), it's essential to have an effective load-balancing strategy. Disproportionate task distribution can hinder cloud infrastructure performance, increasing task completion times and reducing system efficiency. This project aims to develop a robust load-balancing algorithm using the Enhanced Firefly Algorithm (EFA), a Metaheuristic Optimization-based method. EFA is designed to prevent overloading by efficiently allocating tasks among VMs, considering their current loads. This approach aims to improve cloud performance, reduce operational costs, minimize VM migrations, shorten the algorithm's runtime, and enhance the load balancing process's reliability and efficiency.

1.3 Problem Statement

One major challenge in cloud computing (CC) that affects network performance is load balancing. When Virtual Machines (VMs) are assigned tasks that exceed their resource limits, imbalance occurs. The existing techniques, which shift the burden from one virtual machine (VM) to another, usually just move the issue of resource misuse to a different computer. The current study is to present a novel optimization technique designed especially to balance the workload among cloud computing infrastructures.

1.4 Research Question

How could an enhanced version of the Firefly algorithm, designed for optimal distribution of resources in a cloud computing environment, enhance the effectiveness of load balancing?

1.5 Objectives of Research

The objectives of the research focusing on the Enhanced Firefly Algorithm (EFA) for load balancing in cloud computing are as follows:

- **Create an Enhanced Load Balancing Algorithm**: Create a sophisticated load balancing algorithm based on the Enhanced Firefly Algorithm (EFA) that uses Meta heuristic Optimization to effectively distribute tasks in cloud environments.
- **Ensure Even Workload Distribution Across VMs**: Maintain a balanced workload across Virtual Machines (VMs) to avoid overloading and under utilization in the cloud infrastructure.
- **Improve Resource Utilization**: Increase resource efficiency by intelligently allocating tasks based on VM load, thereby optimizing overall system performance and resource usage.

- **Reduce Operational Costs and VM Migrations**: Work to reduce the frequency of VM migrations as well as the operational costs associated with task allocation and load balancing.
- **Improve Cloud System Performance and Reliability**: Increase the performance of the cloud infrastructure by shortening task completion times and ensuring a more reliable and effective load-balancing process.
- **Addressing Current Load Balancing Techniques' Challenges**: Addressing the shortcomings of existing load balancing methods, particularly their proclivity to ignore the current load on VMs, resulting in resource reallocation and imbalance.
- **Innovate in Task Scheduling and Allocation**: Implement novel task scheduling strategies that are more adaptive and efficient, particularly in varying and high-volume workload conditions.

1.6 Outline

Finally, you can now close Section 1 by outlining the structure of the report, for instance: The remainder of the report is organised as follows. Section 2 presents the relevant theory and works closely related to the proposed one. Section 3 describes details of the proposed approach. In Section 4, we describe the techniques that are used to address the problem, as well as all proposed test flows. The implementation and process methods are presented in Section 5, while evaluation results appear in Section 6. Finally, we provide conclusions and discussions of future research directions in Section 7.

2 Related Work

The cloud pro-vides its users with easily expandable and effi-cient services, eliminating the need for them to set up and operate physical infrastructures. As the cloud computing industry grows, it encounters new challenges, one of which is load balancing. A consider-able deal of effort has gone into tackling these issues, resulting in a plethora of new approaches, algorithms, and frameworks. This section will delve into the specifics of current research on Distributed Load and Task Scheduling, Virtual Machine (VM) Allocation, and Resource Allocation Using Optimization Techniques.

2.1 Distributed Load and Task Scheduling in Cloud Computing

Mishra and Tiwari (2020) The research critically addresses the challenge of load balancing in cloud computing, presenting both static and dynamic algorithms aimed at optimizing workload distribution. However, a notable issue emerges in scenarios of demand spikes. The algorithms, while effective in evenly distributing tasks across virtual machines under normal conditions, potentially falter during high-demand periods. This inefficiency is marked by the system's tendency to overload a single virtual machine with tasks, leading to a noticeable increase in task completion time. This highlights a crucial limitation in the algorithms' adaptability and responsiveness to fluctuating demands, underscoring the need for more dynamic and scalable solutions in load balancing within cloud environments. The algorithm developed by Chen et al. (2020) utilized the meta-heuristic whale

optimization algorithm (WOA) to optimize job scheduling in cloud computing environments with limited processing capabilities. They modified the standard WOA to enhance its ability to search for the most effective solutions in task scheduling. This improved method bolstered the utilization efficiency of system resources for both small-scale and large-scale tasks. Prakoso et al. (2021)The research on fuzzy logic for task scheduling effectively improved server utilization and load balancing, particularly in high-load situations, by efficiently distributing HTTP requests. It demonstrated better performance in terms of CPU and RAM usage, achieving high throughput. However, it scored lower on the Fairness Index at 0.45, indicating a less equitable task distribution across servers. This suggests a trade-off between efficiency and fairness. The study primarily assessed performance based on makespan minimization and energy consumption, highlighting a need for a more balanced approach that includes fairness in task allocation. Sahoo et al. (2022)The Modified Crow Search Optimization (MCSO) algorithm outperformed standard algorithms like Genetic Algorithm, Particle Swarm Optimization, and Shuffled Frog Leaping Algorithm in minimizing makespan for task scheduling in heterogeneous multiprocessor systems. MCSO also excelled in speedup, especially for larger population sizes and tasks in dynamic grid computing environments. However, challenges included scalability issues, maintaining consistent performance across diverse tasks, and potentially higher computational overhead. A notable limitation was its local search algorithm's reliance on random selection rather than identifying alternative Virtual Machines (VMs) for task allocation.

2.2 Virtual Machine Allocation in Cloud Computing

Shirvani and Babaeikiadehi (2022)To handle the VM migration in the cloud a novel system architecture for cloud data centers that integrates the WOA with a linear regression model for accurate short-term resource demand forecasting. This hybrid optimization technique is based on the hybrid WOALR model and excelled in reducing SLA violation rates, power, and unnecessary VM migrations by minimizing prediction errors. Additionally, this method achieved maximum resource allocation.

Shen and Chen (2020)The RIAL approach is a load-balancing strategy designed to consider resource utilization intensity. It carefully selects destination physical machines (PMs) for migrating virtual machines (VMs) to minimize post-migration communication overhead. RIAL aims to achieve faster, cost-effective load balancing across the system and reduce future imbalances by considering specific weights in the VM relocation and PM placement process. Tuli and Kaur (2021)The OH-BAC-FUP algorithm, designed for predicting resource utilization in cloud environments, aims to enhance load balancing. Compared with predictive models LR and OPLR, OH-BAC-FUP-OPLR excelled by using less energy, reducing SLA violations, and minimizing VM migrations and host shutdowns. However, it faces challenges in computational complexity and the need for continual adaptation to evolving cloud workloads. N et al. (2023)This research focused on developing an optimized scheduling method for cloud computing to efficiently allocate virtual machines (VMs) and distribute tasks to address load balancing and resource utilization challenges. The proposed solution, D2B-CPU based allocation, utilized the CloudSim tool to simulate various scenarios with varying numbers of VMs and tasks, aiming to enhance performance metrics like execution time and resource balance. Panda et al. (2022)Memory, bandwidth, and CPU utilization were among the new performance indicators introduced in the study. The D2B-CPU method effectively reduced the degree

of imbalance, according to the experimental results.

2.3 Resource Allocation in Cloud Computing

Bo (2022), presented the NA-LB method, designed for allocating cloud computing resources, focuses on load balancing by using an algorithm that tracks Virtual Machine (VM) process parameters through vector values similar to geographical coordinates. It benchmarks resource node distribution, showing enhanced load distribution and increased data processing efficiency in cloud computing clusters. Anjum and Parveen (2022) researcher found that the Overload-Underload (OU) detection algorithm efficiently manages VM live migration, enhancing power and energy efficiency by reducing downtime and total migration time. This is achieved by preemptively assessing the host's load condition to minimize unnecessary migrations. The algorithm significantly improved load usage, and fault tolerance, and reduced the overall makespan. Zhang et al. (2022), developed a Deep Reinforcement Learning (DRL)–based algorithm that effectively decides on task offloading and computing resource allocation, targeting specific optimization goals. It selects the optimal computing node and approaches for each problem, leading to reduced total energy consumption and average task response time, thereby boosting system efficiency.

2.4 Research Gap

The study highlights a significant gap in cloud computing, particularly in task scheduling algorithms, where performance is declining, leading to increased wait times for task execution. This issue stems from the algorithms' focus on optimizing operational metrics like execution time, resource usage, and cost, often neglecting the actual workload demands on Virtual Machines (VMs) crucial for high service quality and balanced performance. Recent efforts to improve VM assignment and resource optimization haven't fully addressed service quality and real-time VM workloads, emphasizing short-term gains over long-term system stability and user satisfaction. Additionally, there's a need for predictive algorithms capable of adapting to current and future VM workloads, essential for preventing overloads and ensuring scalable cloud infrastructure. This calls for a more sophisticated approach in task scheduling that considers a broader range of factors, beyond just efficiency metrics, to maintain system stability and meet user demands effectively.

The Enhanced Firefly Algorithm (EFA) presents a promising solution for the challenges in cloud computing task scheduling, thanks to its capacity for managing complex optimization problems. By intelligently considering both current and future VM loads, EFA ensures more effective load balancing, balancing cost-effectiveness with service quality. Its metaheuristic optimization capabilities allow for dynamic task scheduling, adapting in real time to fluctuating workloads and conditions in the cloud environment. This adaptability helps maintain performance while reducing scheduling delays, positioning EFA as a key contender to address existing gaps in cloud computing task scheduling technologies and research.

3 Methodology

This study aims to refine the brightness mechanism of the conventional Firefly Algorithm to boost load balancing efficiency in cloud computing. The standard Firefly Algorithm operates through a tri-phase process: creating an Initial population, guiding fireflies

based on Attraction and Movement:, and assessing their positions for Evaluation and Adaptation. The Enhanced Firefly Algorithm proposed is intended to decrease the operational time required by Data Centres for processing tasks across their network of Virtual Machines.

3.1 Firefly Algorithm

The Firefly Algorithm (FA) distinguishes itself in meta-heuristic optimization by mimicking fireflies' behavior, excelling in exploring new solutions and refining existing ones through its attraction-based search mechanics, effectively avoiding local optima traps. In comparison, Ant Colony Optimization (ACO) focuses on pheromone trails for solution-building, making it strong in exploiting known paths but slower in convergence and exploration. The Round Robin Algorithm (RRA) is a deterministic scheduling method, lacking FA's dynamic optimization capabilities. FA's simple rules yet computational intensity make it ideal for high-dimensional problems, adaptable to various objectives, and capable of hybridizing with other algorithms for enhanced performance, unlike ACO and RRA. This flexibility positions FA as a superior choice for continuous, high-dimensional, or complex optimization tasks, balancing rapid convergence with thorough exploration. Padmavathi et al. (2020)

The FA contains three phases: Initial Population, Attraction and Movement, and Evaluation and Adaptation.

3.2 Initial Population

A population of fireflies is created, where each firefly represents a potential solution to the problem. The positions of these fireflies are randomly generated within the problem's search space. Each firefly is assigned a brightness level that corresponds to the quality of its solution. This step is crucial as it lays the groundwork for the algorithm to explore the solution space and start the optimization process.

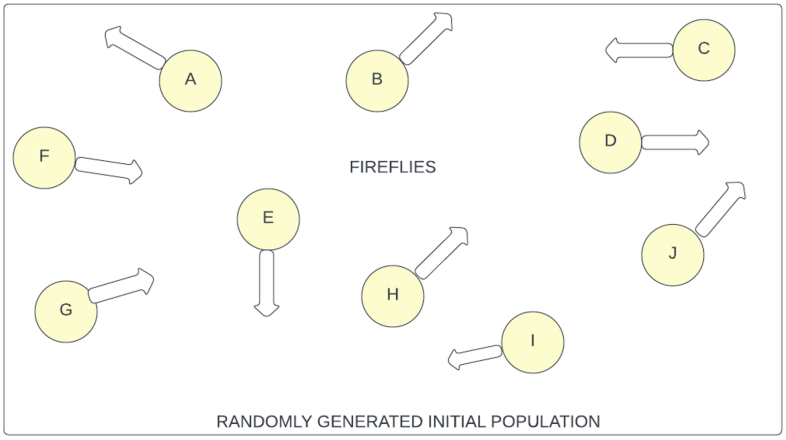

Figure 1:Initial Population.

3.3 Attraction and Movement

Once fireflies explore the solution space it start and gravitate towards each other, with the pull strength diminishing over distance; this means fireflies are more drawn to nearby, brighter peers. They move in a way that combines their attraction to brightness with a degree of randomness to ensure exploration. Following their movement, fireflies update their positions within the search area, each new position symbolizing a potential new solution to the optimization problem.

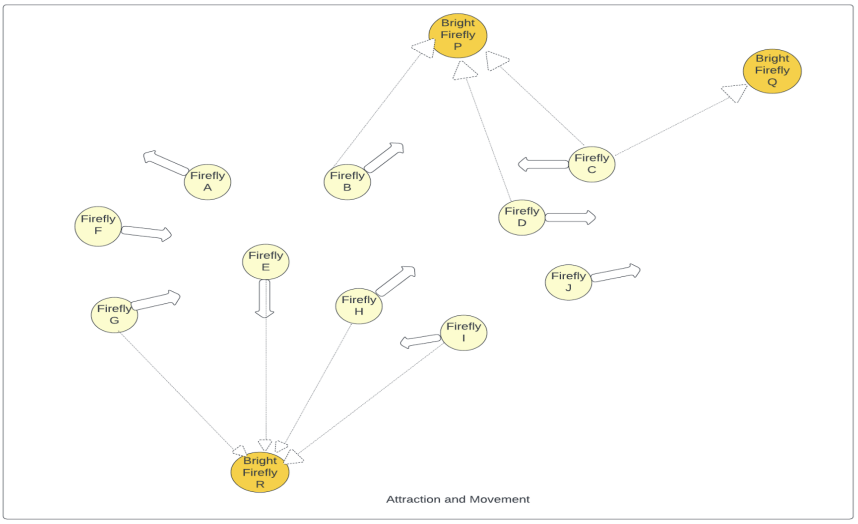

Figure 2.Depicts the attraction and movement.

3.4 Evaluation and Adaptation

After moving, the Firefly Algorithm re-evaluates each firefly's position by assessing their brightness, which indicates the quality of the solution. Fireflies compare their new brightness to the previous one to decide if they've found a better solution. A firefly that discovers a more optimal spot will update both its position and brightness. This iterative process continues, with fireflies either moving to better positions or remaining stationary if no superior solutions are found.

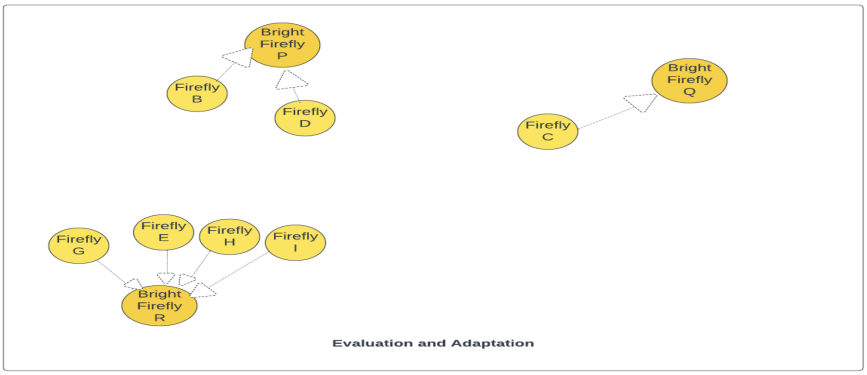

Figure 3.Depicts the Evaluation and Adaptation.

3.5 Termination

The algorithm checks if the termination criteria have been met — this could be a set number of iterations, a time limit, or a satisfactory level of solution quality.

3.6 Process Flowchart

The Firefly Algorithm initiates with setting system parameters and generating an initial population of fireflies, each representing a potential solution, with their brightness indicating solution quality based on the objective function. The brightness of each firefly is reassessed after movement, allowing them to update their position if a better solution is found. This cycle of evaluation and adaptation repeats until convergence criteria, like a specified number of iterations or a target solution quality, are met.

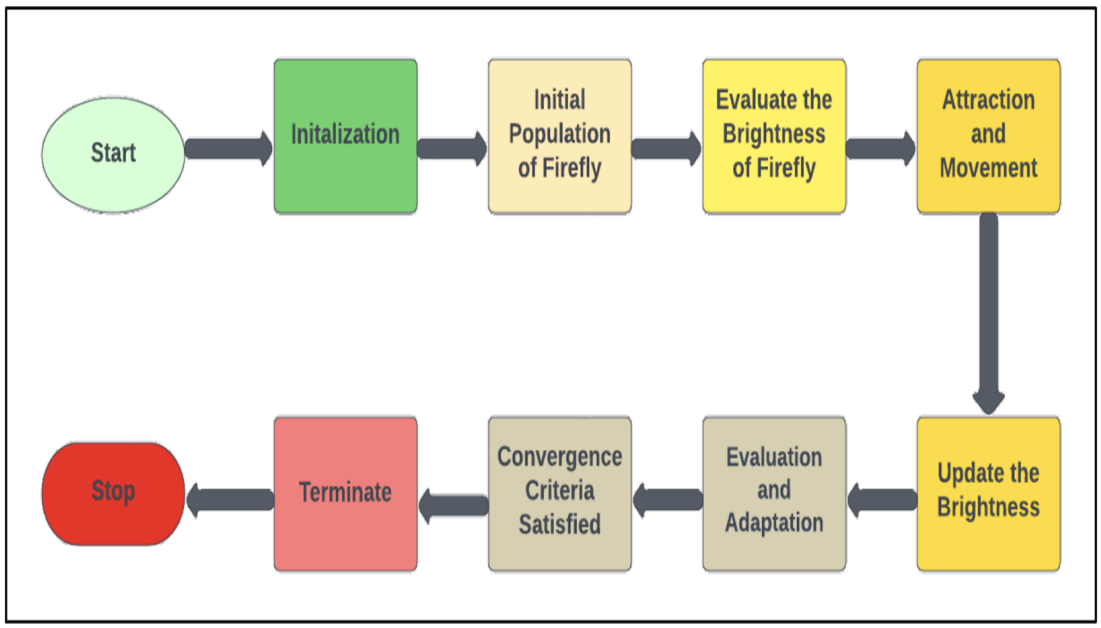

Figure 4.Process Flow Chart

4 Design Specification

The EFA will use the traditional FA method, but the adaptation process will be improved. During the initialization process, the specifications for the DC, VMs, and cloudlets are defined. These specifications are used to create the DC, VMs, and cloudlets that will serve as the EFA's initial population. The EFA improves the optimization process by taking into account the viability of the entire solution during mutation rather than just individual parameters, and it guides the search for an optimal solution from a more global perspective influenced by the brightness (fitness) of each solution. Following the successful creation of the initial population, assign a brightness level to each firefly based on an objective function, such as minimizing load imbalance and maximizing throughput. Following that, the calculation will be based on the attractiveness of each firefly to other fireflies based on their brightness and distance from each other. A typical FA modifies

the brightness parameter of firefly for a VM at random, causing the VM to simulate and update depending on the brightness level of each assigned work. However, because the parameters were not properly allocated, the resulting VM may not function properly. To avoid this, the VM will be modified by calculating the brightness level and replaced with a different VM with the appropriate specifications and calculated value, rather than simply changing the simulation and brightness value. The refined and optimized group of virtual machines (VMs) that emerges from the adaptation and convergence process will be used to set up a cloud computing environment. This setup will include data centers (DCs), virtual machines (VMs), and tasks (cloudlets). The cloudlets will then be processed by the VMs in the DCs in a simulation. The result of this simulation will be the total time required to complete all of the cloudlets' tasks.

4.1 Enhanced Firefly Algorithm

The Enhanced Firefly Algorithm (EFA) is an advanced variant of the traditional Firefly Algorithm (FA), designed to optimize virtual machine (VM) placement in cloud computing setups, with a focus on efficiently managing workload distribution across VMs. The EFA modifies the FA's original three-stage process, particularly the adaptation and convergence stages, to ensure more effective VM deployment within the cloud infrastructure, resulting in improved system performance and resource management.

4.1.1 Initial Population

A starting group or 'population' of elements is created in the initial stage of the Enhanced Firefly Algorithm (EFA). This population contains critical components required to address a particular challenge—in this case, improving load balancing efficiency in a cloud computing environment. Data Centers (DCs), Virtual Machines (VMs), and Cloudlets are critical components for addressing this challenge. A collection of VMs is treated as a swarm to achieve optimal VM placement, with each VM analogous to an individual firefly in the EFA. The figure 5 depicts this initial configuration with the proposed architecture.

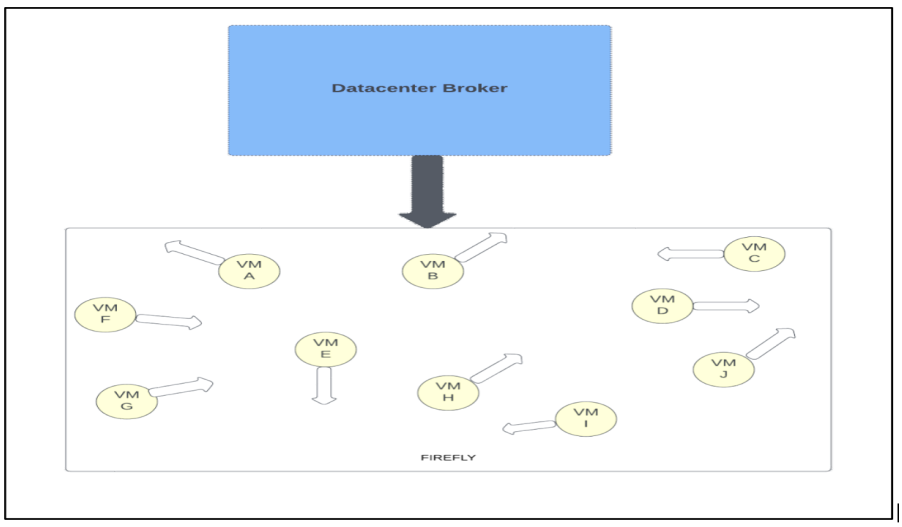

Figure 5.Initial Populations of EFA with Datacentre Broker.

4.1.2 Brightness Function

A key step in the initial population stage of the Enhanced Firefly Algorithm (EFA) is calculating the 'brightness' of each Virtual Machine (VM). This brightness is determined by how well a VM's load stays below a certain maximum capacity threshold, indicating its efficiency and suitability for the task. The VM with the best brightness—meaning it operates most effectively under the threshold—will be chosen as the "best firefly." This best-performing VM sets the bar for the EFA's subsequent steps, guiding the optimization of VM placements.

4.1.3 Attraction and Movement

Based on the previous stage's input, the best-performing Virtual Machine (VM), identified as the 'fittest firefly' based on its brightness, serves as a beacon in the following stage. During this phase, known as 'Attraction and Movement,' other VMs are algorithmically steered toward the brightest one by using an attraction operator. This movement mimics the natural behavior of fireflies being drawn to the most luminous member of their group. The goal is to align the other VMs in the direction of the most optimal VM configuration, as determined by the brightness function. The figure 6 demonstrates how the most efficient VMs are clustered together to improve overall system performance.

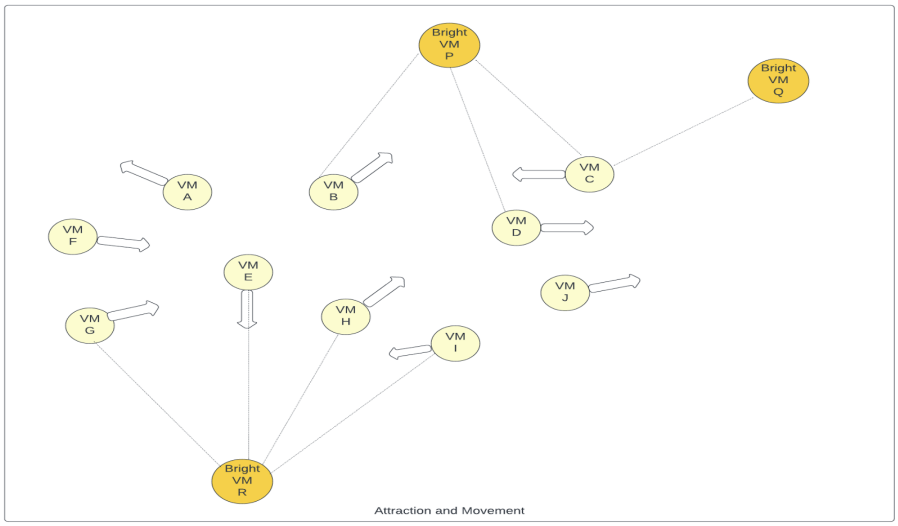

Figure 6.Attraction and Movement of VM's.

4.1.4 Evaluation and Adaptation

In this phase of the Enhanced Firefly Algorithm tailored for cloud computing, the evaluation and adaptation process for the newly attracted Virtual Machine (VM) is carried out with an improved approach. Unlike the traditional evaluation and adaptation method where a random VM is chosen and only one of its parameters is altered—such as CPU capacity, amount of RAM, or storage size—the enhanced process involves a more comprehensive change. This upgraded evaluation and adaptation strategy replaces the entire VM with one that has superior parameters, potentially leading to a more efficient and effective cloud infrastructure. The goal of this enhanced evaluation and adaptation is to significantly uplift the performance characteristics of VMs, thus leading to better optimization of the resources within the cloud environment. The procedure will be improved by replacing the existing virtual machines (VMs) with ones that have superior parameters, with a particular emphasis on enhancing the adaptation process. This process is depicted in the Figure 7.
The load will be distributed across the CC architecture using this evaluation and adaptation list of VMs.

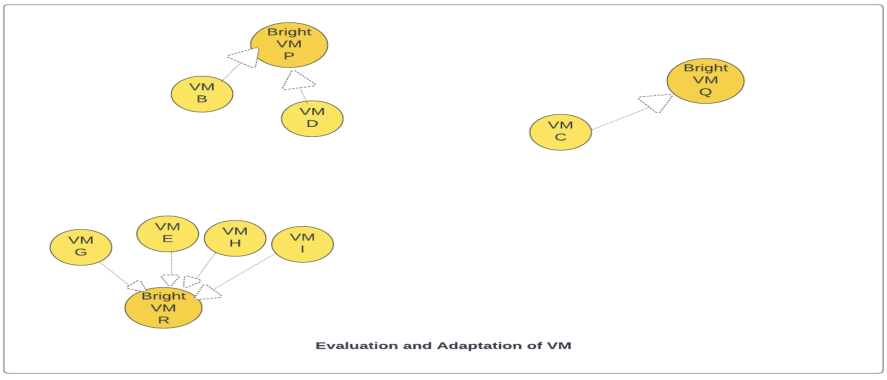

Figure 7.Adaptation of VM's

4.2 Pseudocode

The mathematical model for this algorithm involves several key components:

1. **Firefly Initialization:** Each firefly's position in the d-dimensional space can be represented as a vector: $Xi = (xi1, xi2, ..., xid)$ where i is the index of a particular firefly in the population.

2. **Light Intensity (I):** The light intensity I(x) of a firefly at position x is typically associated with the fitness value calculated from the objective function f(x). For a minimization problem, the light intensity could be inversely proportional to the objective function, whereas for maximization, it is directly proportional.

3. **Attractiveness(β) :** The attractiveness β of a firefly is a function of the light intensity, which decreases with distance r from another firefly. It can be modeled as: $\beta(r) = \beta_0 e^{-\gamma r^2}$ Here,β_0 and $gamma$ the light absorption coefficient that controls how attractiveness decreases with distance.

4. **Distance (r):** The distance r_{ij} between two fireflies i and j in a d-dimensional space can be calculated using the Euclidean distance: The distance between two points is given by $r_{ij} = \sqrt{\sum_{k=1}^{d}(x_{ik} - x_{jk})^2}$.

5. **Movement:** The movement of a less bright firefly i towards a brighter firefly j is determined by: $X_i^{(\text{new})} = X_i^{(\text{old})} + \beta(r_{ij}) \cdot (X_j - X_i) + \alpha \cdot (\text{rand} - \frac{1}{2})$ Here,α is a randomization parameter and *rand* is a random number generator uniformly distributed in [0, 1].

6. **Update Light Intensity:** After the movement, the new solution is evaluated, and the light intensity of the firefly i is updated based on the new position $X_i^{(new)}$.

7. **Algorithm Termination:** The convergence criterion is usually based on the number of epochs (iterations) or if the best solution has not improved over several iterations.

Algorithm 1 Firefly Algorithm

```
0: Begin
0: Initialize the firefly population with random solutions
0: Define light intensity I(x) at x (usually related to the objective function)
0: Define absorption coefficient γ
0: for each epoch do
0:    for each firefly i = 1 to population_size do
0:       for each firefly j = 1 to population size do
0:          if I(i) > I(j) then
0:             Move firefly i towards j in d-dimensional space
0:             Update attractiveness
0:             Evaluate new solutions and update light intensity
0:          end if
0:       end for
0:    end for
0:    Rank the fireflies and find the current best
0:    if convergence criteria is met then
0:       Exit loop
0:    end if
0: end for
0: Output the best firefly
0: End =0
```

The pseudo-code describes the firefly algorithm, which begins by generating a group of fireflies with randomly generated solutions. It assigns a light intensity to each firefly based on the objective function, with a higher intensity indicating a better solution, and sets an absorption coefficient that influences the attractiveness of fireflies to one another. The algorithm then enters a loop in which the intensity of each firefly is compared to all others. If another firefly's intensity is higher, the firefly moves towards it in a multidimensional space, improving its solution by updating its attractiveness and light intensity. Fireflies are ranked at the end of each round, or epoch, to determine the best solution. When this solution meets the predefined convergence criteria, the algorithm terminates; otherwise, it continues until all epochs have been completed. Finally, the algorithm presents the best solution, corresponding to the firefly with the most light intensity.Sababha et al. (2018)

5 Implementation

CloudSim is a simulation tool that models and tests cloud computing scenarios, providing a virtual environment for experimenting with data centers, virtual machines, and resource management. In the context of cloud computing simulation using CloudSim, a data center is the core entity composed of numerous hosts that manage virtual machines (VMs), essentially functioning as an Infrastructure as a Service (IaaS) provider. It processes requests for VM provisioning, typically from data center brokers. The DatacenterBroker class acts on behalf of users to manage two key mechanisms: submitting VM provisioning requests and assigning tasks to VMs, which necessitates users to extend this class to implement custom policies for their experiments. The Host is another critical component that is responsible for VM management tasks such as creation, destruction, and task processing allocation. It operates according to a set policy for allocating resources like memory, processing elements, and bandwidth. Within the host, each VM is a software implementation that emulates a physical machine, partitioning the host's resources to run various tasks. Lastly, the Cloudlet represents these tasks or workloads, encapsulating the computational requirements of applications and managed by the scheduling policy within the DatacenterBroker class. Together, these components interact to simulate a cloud environment where computational tasks are processed and managed efficiently. It enables the evaluation of cloud designs and applications without the need for physical cloud setups, saving time and resources. CloudSim is particularly useful for researching dynamic resource provisioning and load balancing in cloud infrastructures it is depicted in Figure 8.

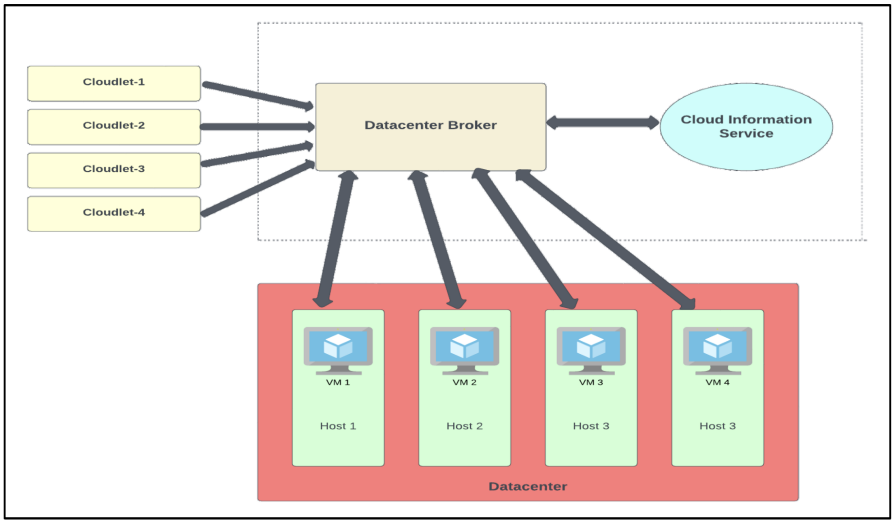

Figure 8 : Proposed Architecture.

5.1 Data center

This functions operate similarly to a physical server and is in charge of generating Virtual Machines (VMs), which are an essential component of cloud computing infrastructure. The VMs created in this data center are tasked with performing basic processing functions. To deploy the Enhanced Firefly Algorithm (EFA), a data center is set up using the specific parameters listed in Table below .

Parameters	Value
Operating System	Linux
VMM	Xen
RAM	2048
Storage	1000000
Bandwidth	10000
Architecture	X86
Cost	3.0
Cost of Storage	0.001
Cost of Bandwidth	0.01
Cost of Memory (per size)	0.05
Time zone	10.0

5.2 Virtual Machine

Within a Data Center (DC), Virtual Machines (VMs) are created to emulate the functionalities of physical machines using the infrastructure's resources. These VMs, operating within the DC, replicate the computational architecture of actual servers and are tasked with handling assigned operations. Parameters for configuring these VMs in a simulated environment like CloudSim are specified in the Table below.

Parameters	Value
VMM	Xen
RAM	512
SIZE	10000
Bandwidth	1000
Mips	1000
Psenumber	1

5.3 Cloudlets

In CloudSim, a cloudlet represents a computational task or job that is allocated to a Virtual Machine (VM) for execution. The term 'length' regarding a cloudlet refers to the total instruction count that the cloudlet must process to completion. There's a designated table, referred to as the Table below, which details the various attributes and settings that define the characteristics of a cloudlet within the simulation environment.

Parameters	Value
Length	40000
File size	300
Output size	300
Psenumber	1

5.4 Scale of Experiments:

The Enhanced Firefly Algorithm (EFA), along with established methods such as the Round-Robin (RRA) and Ant Colony Optimization (ACO), will be implemented within a CloudSim environment to determine the most efficient virtual machines (VMs) for data centers (DCs) across a range of experiments. A total of 15 experiments were mentioned, with 5 utilizing EFA, and the remaining 10 employing RRA and ACO each. The experiments varied in the number of cloudlets to demonstrate the scalability and adaptability of the algorithms.

6 Evaluation

In the CloudSim framework, a cloud architecture is configured to run three separate simulation scenarios. The first scenario employs the Enhanced Firefly Algorithm (EFA) to

distribute a variable number of cloudlets and achieve load balance within the architecture. The second scenario uses the Round Robin Algorithm (RRA) to evenly allocate loads, considering different sizes of cloudlets across the proposed structure. Additionally, the Ant Colony Optimization (ACO) algorithm is applied as another load balancing technique, leveraging ant behavior models to enhance task distribution to Virtual Machines (VMs), thus optimizing overall load management. The outcomes from these simulations are methodically examined and contrasted to assess the effectiveness of each load-balancing strategy.

6.1 Experiment 1

The figure 9 compares the time required by different algorithms to complete a set of tasks (cloudlets) in a data center (DC). When compared to Ant Colony Optimization (ACO) and Round Robin Algorithm (RRA), the Enhanced Firefly Algorithm (EFA) performs better in completing tasks (cloudlets) in a data center. Its main reason for this is its strategy of allocating the most capable virtual machines, termed "Brightest VMs," for cloudlet processing. This targeted allocation ensures that the most efficient resources are used to complete tasks as quickly as possible. EFA dynamically adapts to varying workloads and VM performance, in contrast to ACO, which relies on probabilistic techniques for pathfinding and may not optimally utilize individual VM capabilities, or RRA, which evenly distributes tasks without considering VM performance. his flexibility and focus on maximizing the efficiency of each VM contribute significantly to its faster processing times, making EFA a more effective choice for managing tasks in data centers.

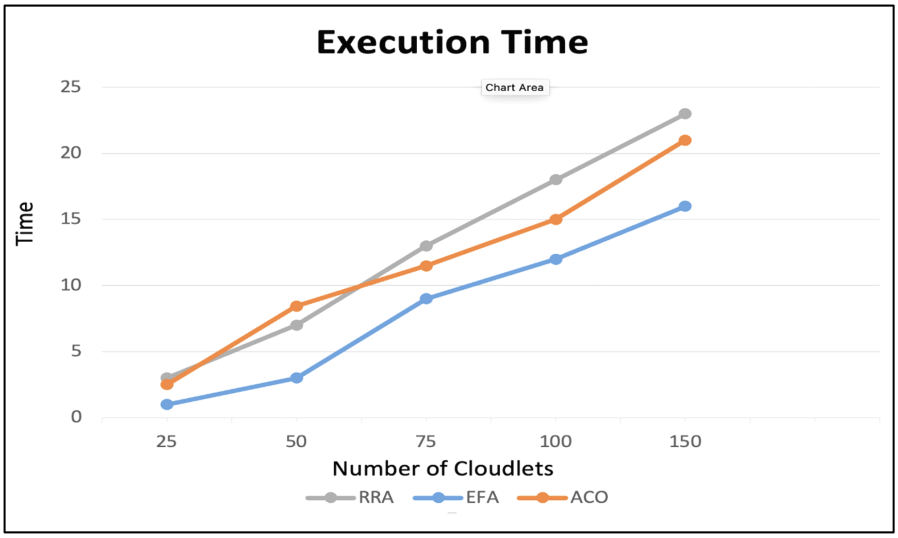

Figure 9 : Execution Time .

6.2 Experiment 2

In data center environments, the Enhanced Firefly Algorithm (EFA) outperforms the Round Robin Algorithm (RRA) and Ant Colony Optimization (ACO) in terms of resource allocation efficiency. This is especially noticeable in its ability to reduce the total time required to complete all cloudlets. EFA accomplishes this by assigning tasks to the most capable virtual machines (VMs), dubbed "Brightest VMs." Unlike RRA's cyclic task allocation and ACO's probabilistic pathfinding, EFA's strategy maximizes the use of each VM's capabilities. In EFA, total resource utilization is more efficient, as calculated by multiplying the total execution time of cloudlets by CPU utilization time. This results in improved performance. This efficiency becomes even more pronounced as the number of cloudlets increases, demonstrating EFA's superior scalability and adaptability in handling variable and intensive workloads, a crucial factor in modern data center operations, as shown in Figure 10.

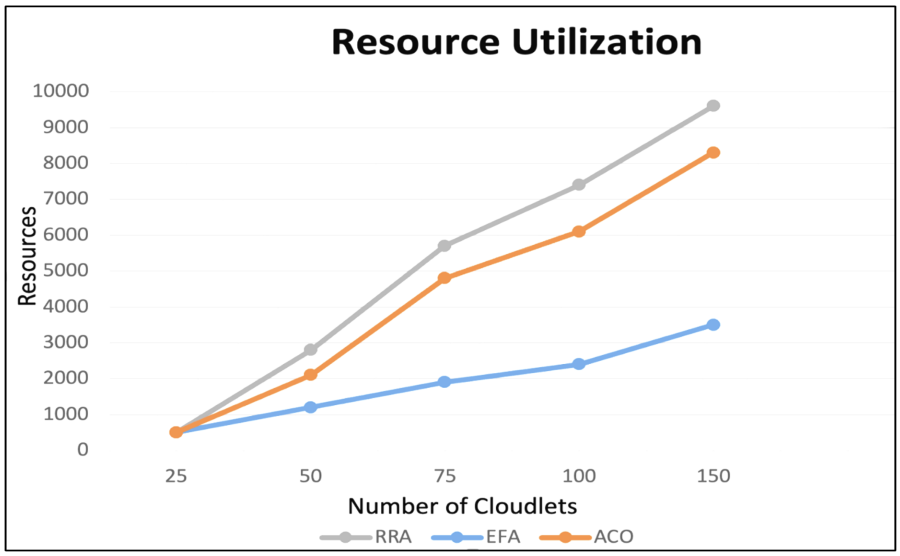

Figure 10 :Resource Utilization.

6.3 Experiment 3

Figure 11 shows that in terms of energy conservation during simulations, the Enhanced Firefly Algorithm (EFA) outperforms both the Round Robin Algorithm (RRA) and the Ant Colony Optimization (ACO). This efficiency is largely due to EFA's optimized resource allocation strategy, which involves assigning tasks to the most capable and energy-efficient virtual machines (VMs), referred to as "Brightest VMs." This targeted allocation ensures that resources are used more efficiently, resulting in significant energy savings. In contrast, RRA's equal task distribution, regardless of VM capabilities, can result in inefficient energy use, and ACO, while adept at finding optimal paths, does not specifically target energy efficiency. Consequently, EFA demonstrates a more significant reduction in energy usage, particularly under varying and intensive workloads, making it a more sustainable and cost-effective choice for data center operations compared to its counterparts.

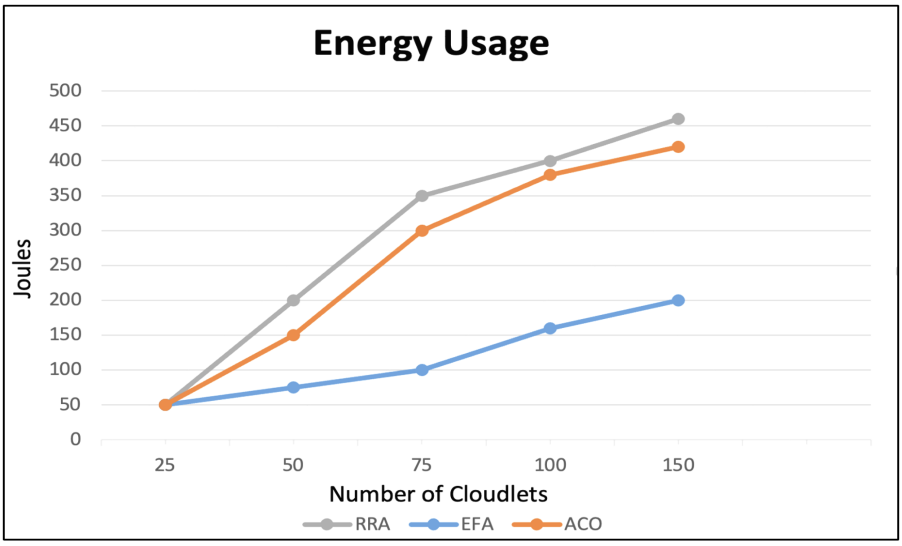

Figure 11 : Energy Usage.

6.4 Discussion

The outcomes of the research demonstrate that the Enhanced Firefly Algorithm (EFA) is superior for load balancing in cloud computing (CC) when compared to the Round Robin Algorithm (RRA) and the Ant Colony Optimization (ACO) algorithm, both of which are established algorithms in the field. The data shows that as the number of cloudlets increases, the EFA maintains better efficiency, exhibiting a smaller rise in execution time, resource use, and energy consumption than what is observed with the RRA and ACO algorithms. When compared to the enhanced firefly algorithm EFA, the round robin algorithm RRA and ant colony optimization ACO may not allocate resources as efficiently. RRA distributes tasks equally but without considering the varying capabilities of virtual machines, which can result in sub-optimal utilization and increased execution time.

Due to its heuristic nature, ACO, with a more adaptive nature than RRA, may not converge as quickly or as effectively to the optimal solution, potentially resulting in less efficient resource use and higher energy consumption. EFA, on the other hand, directly targets the most efficient allocation of tasks to the best performing machines, resulting in improved load balancing, faster execution times, and lower energy consumption. Due to their inefficient task allocation strategies, the Round Robin Algorithm (RRA) and Ant Colony Optimization (ACO) tend to increase execution time, use more resources, and consume more energy.

The Enhanced Firefly Algorithm (EFA), on the other hand, significantly improves virtual machine (VM) efficiency by employing a brightness function for intelligent adaptation. This function chooses and combines the best VMs before making random adjustments to improve their performance. As a result, EFA is more effective at evenly distributing workload across various volumes of cloudlets. Through the use of EFA, this study achieves its goal of improving load balancing in cloud computing (CC) by reducing execution time, resource utilization, and energy consumption.

7 Conclusion and Future Work

The Firefly Algorithm (FA) is used in this study to develop the Enhanced Firefly Algorithm (EFA), which aims to optimize load balancing in cloud computing (CC). Data centers (DCs), virtual machines (VMs), and cloudlets are created with specific parameters during the initial phase. The EFA uses a brightness function to identify and merge the most appropriate fireflies from the VM population. To improve performance, some VMs are replaced at random with newly created ones. When compared to well-known algorithms such as ACO and RRA, simulations using EFA show superior load balancing, with reductions in execution time, resource usage, and energy consumption.

The future potential for improving the Enhanced Firefly Algorithm (EFA) in the context of cloud computing is vast. There is still room to investigate the EFA's potential for improving fault tolerance in order to maintain service continuity even when individual components fail. Furthermore, the EFA could be extended to dynamically allocate resources for dependent tasks, which is especially important for workflows requiring synchronized execution. The algorithm could be improved to further optimize energy consumption, helping to advance green computing practices. By addressing these issues, the EFA can grow into a more comprehensive solution for complex cloud computing problems.

References

Alameen, A. and Gupta, A. (2020). Fitness rate-based rider optimization enabled for optimal task scheduling in cloud, *Information Security Journal: A Global Perspective* **29**(6): 310–326.
URL: *https://doi.org/10.1080/19393555.2020.1769780*

Anjum, A. and Parveen, A. (2022). A dynamic approach for live virtual machine migration using ou detection algorithm, *2022 6th International Conference on Computing Methodologies and Communication (ICCMC)*, pp. 1092–1097.

Bo, Y. (2022). Cloud computing resource node allocation algorithm based on load balancing strategy, *2022 IEEE 6th Information Technology and Mechatronics Engineering Conference (ITOEC)*, Vol. 6, pp. 1721–1725.

Chen, X., Cheng, L., Liu, C., Liu, Q., Liu, J., Mao, Y. and Murphy, J. (2020). A woa-based optimization approach for task scheduling in cloud computing systems, *IEEE Systems Journal* **14**(3): 3117–3128.

Mishra, A. and Tiwari, D. (2020). A proficient load balancing using priority algorithm in cloud computing, *2020 IEEE International Conference on Machine Learning and Applied Network Technologies (ICMLANT)*, pp. 1–6.

N, K., Yarava, R. K., M S, K. and S, K. (2023). Dynamic virtual machine allocation and load balancing methods for heterogeneous task allocation in public cloud, *2023 International Conference on Communication, Circuits, and Systems (IC3S)*, pp. 1–5.

Ojha, S. K., Rai, H. and Nazarov, A. (2020). Optimal load balancing in three level cloud computing using osmotic hybrid and firefly algorithm, *2020 International Conference Engineering and Telecommunication (EnT)*, pp. 1–5.

Padmavathi, M., Basha, S. M. and Krishnaiah, V. V. J. R. (2020). Load balancing algorithm to reduce make span in cloud computing by enhanced firefly approach, *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*, pp. 896–900.

Panda, S. K., Ramesh, K., Indraneel, K., Ramu, M. and Damayanthi, N. N. (2022). Novel service broker and load balancing policies for cloudsim-based visual modeller, *2022 Seventh International Conference on Parallel, Distributed and Grid Computing (PDGC)*, pp. 232–237.

Perepelkin, D., Ivanchikova, M. and Nguyen, T. (2023). Research of multipath routing and load balancing processes in software defined networks based on bird migration algorithm, *2023 International Russian Smart Industry Conference (SmartIndustryCon)*, pp. 247–252.

Perepelkin, D. and Nguyen, T. (2022). Research of multipath routing and load balancing processes in software defined networks based on artificial bee colony algorithm, *2022 ELEKTRO (ELEKTRO)*, pp. 1–6.

Prakoso, I. A., Hertiana, S. N. and Dewanta, F. (2021). Analysis of fuzzy logic algorithm for load balancing in sdn, *2021 4th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, pp. 401–406.

Sababha, M., Zohdy, M. and Kafafy, M. (2018). The enhanced firefly algorithm based on modified exploitation and exploration mechanism, *Electronics* **7**(8).
URL: *https://www.mdpi.com/2079-9292/7/8/132*

Sahoo, R. M., Padhy, S. K. and Debasis, K. (2022). A new dynamic method of multiprocessor scheduling using modified crow search optimization, *2022 2nd International Conference on Artificial Intelligence and Signal Processing (AISP)*, pp. 1–6.

Shen, H. and Chen, L. (2020). A resource usage intensity aware load balancing method for virtual machine migration in cloud datacenters, *IEEE Transactions on Cloud Computing* **8**(1): 17–31.

Shirvani, M. H. and Babaeikiadehi, S. (2022). A hybrid meta-heuristic-based linear regression algorithm for live virtual machine migration in cloud datacenters, *2022 International Conference on Electrical, Computer and Energy Technologies (ICECET)*, pp. 1–5.

Tuli, K. and Kaur, A. (2021). Load balancing scheme for optimization of virtual machine migration using swarm in cloud environment, *2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, pp. 1–6.

Zhang, Y., Zhu, H., Tang, D., Zhou, T. and Gui, Y. (2022). Dynamic job shop scheduling based on deep reinforcement learning for multi-agent manufacturing systems, *Robotics and Computer-Integrated Manufacturing* **78**: 102412.
URL: *https://www.sciencedirect.com/science/article/pii/S0736584522000977*