Workload Scheduling Optimization using Dispersive Fly Optimisation in Cloud Paradigm

Yuvaraj Gandhi S[1] & Revathi T[2]
[1]*Department of Computer Applications, PSG College of Arts & Science, Coimbatore, India*
[2]*Department of Computer Science, PSG College of Arts & Science, Coimbatore, India*

Abstract*—Virtual Machine (VM) integration methods use two vital technical aspects, virtual machine communication and virtual machine I/O interception, to optimise load balancing in cloud data. Virtual Machine communication facilitates the seamless input and output data stream flow between software packages executed within individual virtual machines. Virtual Machine I/O interception mechanisms play a crucial role in redirecting the input and output of application packages. This virtual machine communication and virtual machine i/o interception uses resource utilisation and resource allocation on the cloud network to make efficient task scheduling by designing, optimising, and securing workload scheduling. This will be achieved by balancing cost-effectiveness, maximising resource utilisation, ensuring top-notch service quality, avoiding service level agreement violations, and optimising workload performance, which is crucial for achieving operational excellence. Many research articles do not concentrate on these challenges. The limited resource-level provisioning is causing difficulties in obtaining accurate and detailed workload fluctuations, described by multiple layers of noise. The Long-term Potentiation (LTP) model is vital in signal transmission between two neurons. This research presents a hybrid model using neuroscience, including Dispersive Fly Optimisation and transcranial random noise stimulation (tRNS) ("NDFO-tRNS") for dynamic workload provisioning in a cloud network. The proposed NDFO- tRNS framework forecasts cloud resource utilisation. Finally, the LTP module simulates and predicts the VM workload. The proposed model efficiently integrates resource utilisation and workload scheduling in a cloud network.

***Index Terms*—Cloud Network; Dispersive Fly Optimisation; Resource Allocation; Virtual Machine Integration; Workload Scheduling;**

I. INTRODUCTION

Virtual machine (VM) integration methods are crucial for optimising load balancing in cloud data environments. These methods rely on two critical technical aspects: virtual machine communication and virtual machine I/O interception [1]. Virtual machine communication ensures seamless input and output data flow between software packages executed within individual virtual machines. This facilitates efficient data transfer and communication processes, contributing to overall system performance and resource utilisation. Additionally, virtual machine I/O interception is vital in monitoring and managing input/output operations, enhancing the ability to control and optimise data flow within the virtualised environment. These integrated methods are essential for maintaining a well-balanced and efficient cloud computing infrastructure [39].

The interception mechanisms for Virtual Machine I/O are essential for redirecting the input and output of application packages within a cloud-computing environment. These mechanisms are designed to efficiently use the cloud network's resources to schedule tasks effectively [25]. By employing intent workload scheduling, workload optimisation, and workload security measures, these interception mechanisms ensure that the input and output of application packages are managed and processed to maximise resource utilisation and allocation.

Achieving our goals will depend on balancing cost-effectiveness and maximising resource utilisation while maintaining top-notch service quality and avoiding violations of service level agreements. Furthermore, optimising workload performance is crucial for achieving operational excellence. It is worth noting that a significant number of research

articles fail to address the challenges associated with limited resource-level provisioning, which leads to difficulties in accurately capturing and detailing workload fluctuations [2].

Multiple layers of noise can make it challenging to obtain precise data, often leading to obscured fluctuations. In this context, the Long-Term Potentiation (LTP) model facilitates signal transmission between two neurons. This research introduces a sophisticated hybrid model that leverages insights from neuroscience, incorporating Dispersive Fly Optimization and transcranial random noise stimulation (tRNS) to create the "NDFO- tRNS" framework. This framework is specifically designed for dynamic workload provisioning in a cloud network, focusing on forecasting cloud resource utilisation. Additionally, the LTP module is employed to simulate and predict the workload of virtual machines (VMs). Integrating these components results in an efficient model that effectively manages resource utilisation and workload scheduling within a cloud network [12][15].

The proposed model, a novel approach, operates on three distinct levels.

1. In the initial stage, employ a 'long-term potentiation (LTP) model' for workload scheduling in the cloud network to address the efficient distribution of workloads, an essential aspect of cloud computing.
2. The second stage utilises a Dispersive Fly Optimisation for Handling Workload balance Using Swarm Intelligence Techniques.
3. transcranial Random Noise Stimulation (tRNS) manages time and scheduling efficiency.

The remaining chapter discusses as follows: Chapter 1 discusses an Introduction to Workload Scheduling Optimization; Chapter 2 discusses various literature reviews on Workload Scheduling Optimization using Dispersive Fly Optimisation in the Cloud Paradigm; and Chapter 3 discusses the proposed system. Chapter 4 discusses the results of Workload Scheduling Optimization with various algorithms, and Chapter 5 discusses the result with appropriate outcome research along with the future improvement that makes something more agreeable

II. LITERATURE REVIEW

Song et al. (2024) discussed Artificial synapses to explore the application of IGZO in two terminal artificial synaptic devices [17][22]. Their proposed system recognised the handwritten digits with good realisation based on the synaptic weight, resulting in a higher recognition rate. The proposed system emulated paired-pulse facilitation (PPF), long-term potentiation (LTP), and long-term depression (LTD) for higher recognition rate stimulation are achieved. The research given at the end of this Long-term Potentiation Model for Workload Scheduling in Cloud Network section is one we have taken as a Long-term Potentiation Model Stage for our research. This research is what we have to do in the best possible way for workload scheduling. The constructive result of this research is very supportive of our research. We considered the study's insights and measured the changes through subsequent leaps, refining our research.

X. Wang et al. (2024) proposed the synaptic strength in the existing Bienenstock-Cooper Munro (BCM) learning rule. Threshold-based LTD analyses the synaptic long-term depression (LTD) [39]. The existing BCM learning rules can be overcome by using fixed rule parameters. This research article proposes the dual-threshold BCM learning rule to regulate the reservoir internal connection weights of the echo-state network (ESN) using optimal LTD thresholds. Based on the learning rule made at the existing Bienenstock-Cooper-Munro (BCM) of the research and the LTD analyses, we motivated new ideas to quickly bring our research to the Virtual Machine I/O Interception (synapse) field and Virtual Machine (VM) Integration (Patterns of stimulation) it as best possible. The primary motivation of this research is that the emerging Output Data Stream Flow and Input Data Stream Flow objective of our study is to provide an agronomic outcome to the research, which is a stimulus for the various types of Long-term potentiation model stages.

L. Tian et al. (2023) concentrate on the micro-coil stimulation process's signal-to-noise ratio (SNR) [22]. The resonance WPT theory estimates the micro-coil's transmission efficiency and mutual inductance. Most recently, results from biological experiments show the synaptic plasticity long-term potentiation (LTP). This research took advantage of the study's insights. It

measured the changes through the subsequent leaps, refining our research by getting every impression that gave us general feedback relevant to our study. The main point of this research is that our emerging objective is to provide an agronomic answer to the research, which becomes an impetus for further research into different types of answer-based research.

Hsu et al. (2023) discussed the Optoelectronic Synaptic Memristor for Neuromorphic Computing [5]. Their research, Synapses, shows that it plays a good role in human brain memory and computing. They enhanced a ZTO-based optoelectronic synaptic memristor (OSM) based Hebbian learning for better recognition accuracy. Additionally, the results of biological experiments showed the long-term potentiation (LTP) of synaptic plasticity. Our research aims to analyse and understand the implications of this study on our research. The study provides valuable insights and findings that will contribute to furthering our research beyond conventional perspectives.

Chakraborty et al. (2024) discussed the intelligent Internet of Things (IoT) devices for developing various computing paradigms to address challenges such as latency and resource allocation [7]. This research introduces a framework for intelligent offloading mechanisms and efficient resource allocation in an integrated cloud-fog-IoT environment. The proposed Fuzzy-based Harris Hawks-Genetic Algorithm (HHGA) outperforms existing optimisation algorithms, improving service cost, time, and energy consumption. We used the Quadruple goal optimisation for resource allocation using the LTP model for analyses to drive new ideas for applying our research to Virtual Machine I/O Interception and Virtual Machine Integration. Our Dispersive Fly Optimisation aims to achieve workload balance outcomes for various stages of the long-term potentiation model.

Ullah et al. (2024) discussed that the distribution of tasks in cloud data centres, especially in the infrastructure-as-a-service (IaaS) cloud paradigm, is a persistent challenge [37]. To address this challenge, they propose an improved load-balancing approach called the Hybrid BAT and ABC (HBABC) algorithm, which dynamically distributes resources, resulting in improved accuracy in task allocation and a decrease in the energy level of the cloud data centre. This research suggested future use as a prediction strategy for the resource management system in cloud data centres [24]. The research is provided in the load-balancing approach for the Workload Scheduling of the Cloud Network section. The constructive outcomes of this research have significantly influenced our approach. By incorporating the insights from this research, we have made substantial strides in refining our work.

Liu et al. (2024) discussed the balanced leader election algorithm based on replica distribution in the Kubernetes cluster [6] [21] [22] [26]. This article is an open-source project that provides an orchestration platform for containerised balanced leader election applications. Redundant deployment creates multiple replicas for each application to ensure high scalability and availability. However, this can lead to too many leaders on some nodes, reducing system performance. The proposed balanced leader election algorithm based on replica distribution inspired us to make the Dispersive Fly Optimization (DFO) of imbalanced data against the evenly distributed leaders among worker nodes, solving the performance degradation issue. Realising the importance of the above research and in our article, we have taken the maximum variety of output-based, single-balanced leader elections obtained from cloud data; from that, we have concluded our research in the best way. These factors helped us as they varied according to our publications and the changes we took from the existing research paper. Awan et al. (2024) proposed a deadline-aware fault-tolerant scheduler for cloud computing. The proposed job scheduling in the cloud is crucial for optimising operational costs in data centres [2]. To rectify the problem, tasks can be prioritised based on their deadlines and lengths, and resources can be provisioned and released as needed. Our research is inspired by the fault tolerance mechanism using hybrid replication and the re-submission method. Our research can handle execution hardware failures on the new scheduler, which dynamically acquires and schedules resources for tasks while providing fault tolerance and reducing costs.

Lin et al. (2024) proposed a novel virtual machine consolidation algorithm for handling the resources from cloud computing users [25]. This research proposed a Two-Level Heuristics Virtual Machine Consolidation (TLHVMC) method that considers the overheads of Power Mode Transitions (PMTs). TLHVMC employs a Greedy Strategy-based hybrid heuristic Placement algorithm for resource allocation and implements Time Window-based Centralized

power mode Control policies to manage host power modes. Our Experimental outcome process is framed with the help of TLHVMC. Our research outcome significantly improves data centre energy efficiency and maintains cloud system performance compared to other VM consolidation algorithms.

Shamsa et al. (2024) proposed a distributed load-balancing method for IoT/Fog/Cloud environments with volatile resource support [34]. The research describes a load-balancing method for integrating local schedulers to predict workload and account for volatile mobile nodes as dynamic resources. To simplify the system, it transforms the environment into a grid of equally sized cells and uses an exhaustive search to transfer extra workflows from overloaded to under-loaded. This research motivates us to improve the Extensive software simulations, and our proposed method outperforms others in terms of job completion rate, workload variances, and time-related parameters, with better results.

III. PROPOSED SYSTEM

The rapid growth of cloud computing has sparked a crucial focus on cost reduction in cloud computing platforms, primarily through migrating virtual machines (VM). This article discusses optimised workload scheduling in the Cloud Paradigm. A quadruple-goal optimisation for resource allocation is established, which considers resource consumption between virtual machines (VM) [3]. In the initial phase, this research implements the 'long-term potentiation (LTP) model' for workload scheduling in the cloud network to address the efficient distribution of workloads, an essential component of cloud computing. The subsequent phase involves the utilisation of Dispersive Fly Optimization for Handling Workload balance Using Swarm Intelligence Techniques. Finally, transcranial random noise stimulation (tRNS) oversees time and scheduling efficiency [13].

3.1. Long-term potentiation model for Workload Scheduling in Cloud network

Long-term potentiation (LTP) is a neurophysiological process that involves the persistent strengthening of synaptic connections. This phenomenon is induced by a brief high-frequency presynaptic activity, resulting in increased signal transmission efficiency between neurons. LTP is a fundamental mechanism underlying learning and memory formation in the brain, and its detailed study has significant implications for understanding various neurological and cognitive disorders [26].

3.1.1. Long-term potentiation model stages

Our research proposed the long-term potentiation (LTP) model for workload scheduling in the cloud network to inspire this neurological and mental process [27]. Long-term potentiation (LTP) is a process that involves the persistent strengthening of resource allocation. This phenomenon is induced by a brief period of high- Workload Scheduling activity, resulting in an increased efficiency of resource utilisation and resource allocation between cloud data. LTP is a fundamental mechanism underlying learning and optimising workload performance formation in Resource-level Provisioning [28].

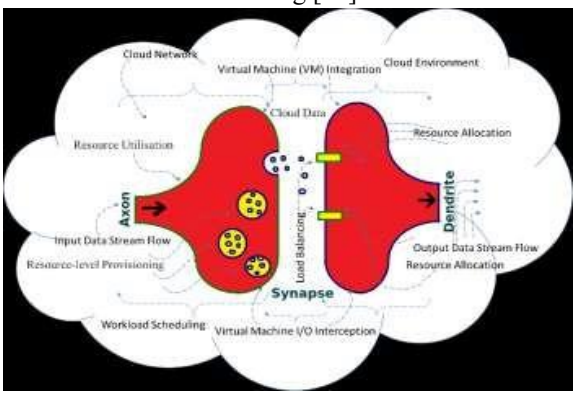

Figure 1. Stage 1 about Input Data Stream Flow

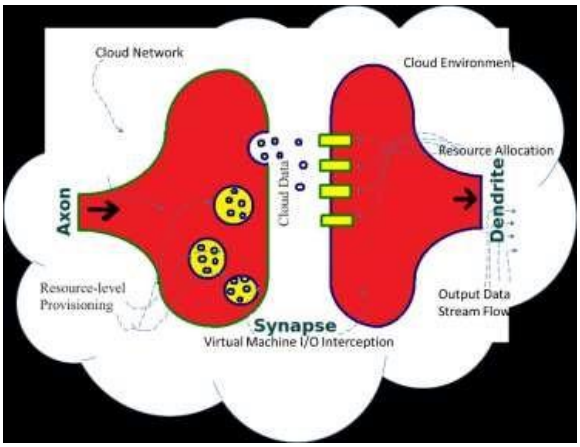

Figure 2. Stage 2 about Virtual Machine I/O Interception (synapse)

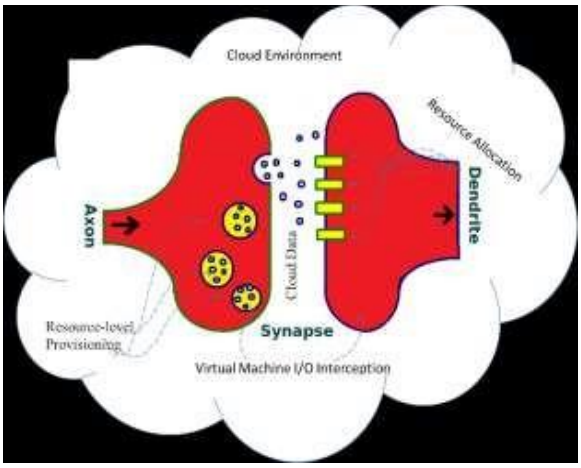

Figure 3. Stage 3 about Virtual Machine (VM) Integration (Patterns of stimulation)

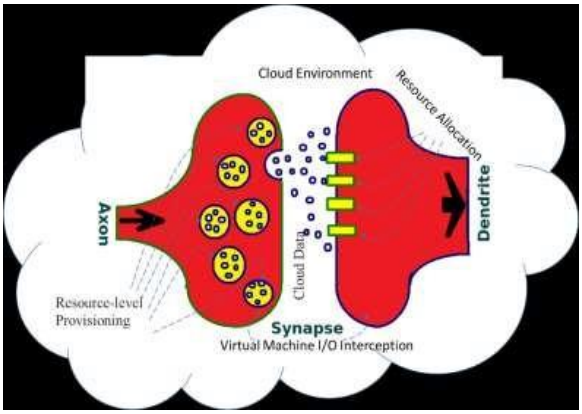

Figure 4. Stage 4 about Output Data Stream Flow

Stage 1: Input Data Stream Flow

Input Data Stream Flow illustrates the process of data handling using Apache Kafka (Figure 1). It connects to the source data and frames the workload via Change Streaming data [29]. The data then flows into Load Balancing Processing, which undergoes continuous processing through aggregation, filtering, and utilising Virtual Machine (VM) Integration. The results become a reality in Virtual Machine I/O Interception. This setup highlights the continuous Input Data Stream Flow role in handling and processing data streams within the Long-term potentiation (LTP) environment. This long-term potentiation (LTP) is fixed as a long-lasting increase in virtual machine I/O interception of resource allocation [4].

Stage 2: Virtual Machine I/O Interception (synapse):

The objective of Virtual Machine I/O Interception is to identify the block number at which the Virtual Machine is conducting read or write operations and potentially halt these operations if the user does not intend to read from or write to that specific block (Figure 2). While direct disk write operations can be interrupted, modifications made through the Virtual Machine can still be observed [15][16]. At the same time, restarting the virtual machine will revert these modifications, indicating possible buffered IO behaviour. The synapse process will be lost upon resuming the virtual machine, indicating possible buffered IO behaviour. Virtual Machine I/O Interception (synapse) system that transmits data from one resource to a free resource by unblocking the user would like to read/write.

Stage 3: Virtual Machine (VM) Integration (Patterns of stimulation)

The Virtual Machine (VM) Integration is compounded with the VM groups and allocation of work and resources by inspiring stimulation patterns (Figure 3). This learning produces optimal solutions for user evaluation data [17]. For this concern, Virtual Machine (VM) Integration collaborates with the three-step process for stimulation patterns. The first step is activating integration, including Axon, resource-level provisioning, and input data stream flow. It also involves configuration and polling, encompassing configuration workload scheduling and polling cloud data. Finally, resources were allocated using region identification, availability, instance type, and instance ID.

Stage 4: Output Data Stream Flow

In long-term potentiation (LTP), synaptic transmitted output data stream flow is recognised as data storage for allocated resource information (Figure 4) [23].

3.1.2. Quadruple-goal optimisation for resource allocation using the LTP model

From the above constraint, neuroscience-inspired long-term potentiation (LTP) helps schedule the workload in a cloud network. This was proposed to maximise resource allocation and job distribution over time by emulating the synaptic plasticity principles observed in organic neural networks. This model uses past workload patterns and feedback systems to dynamically change scheduling decisions, resulting in better resource usage and system performance [5].

The LTP-based workload scheduling model follows a systematic process that includes historical workload

data, synaptic weight adjustment, dynamic resource allocation, feedback mechanisms, and performance evaluation. Initially, historical workload data D is pre-processed to verify data quality and normalise characteristics. Each workload sample d_i is represented as

t a feature vector. $v_j = [v_{j1}, v_{j2}, \ldots, v_{jm}]$

With contextual information as additional features.

Synaptic weights $W = [w_1, w_2, \ldots, w_m]^t$ Between tasks and resources are initialised and adjusted using the Hebbian learning rule as per Equation 1.

$$\Delta W_{jk} = \mu \times v_j \times v_k \quad (1)$$

Here, ΔW_{jk} Represents the change in synaptic weight, μ signifies the learning rate, and $v_j$and v_k Indicate the activities of task j and resource k, respectively. The compatibility of a new task t_{new} With each resource, j is computed using the dot product, resulting in dynamic resource allocation (Equation 2).

$$Compatibility\ (t_{new}, k) = v_{new}.\ w_k \quad (2)$$

This compatibility score directs the allocation of t_{new} To the resource with the highest score. Feedback on task completion timings t_{comp}It has been obtained, prompting synaptic weight adjustments to decrease the error between projected and actual completion times (Equation 3).

$$\Delta Wjk = \mu \times (tcomp - tpred) \times vj \times vk \quad (3)$$

Here, t_{pred}Denotes the projected completion time. Finally, the model's performance is assessed using task completion time, resource use, and system throughput measures. This complete methodology enables the LTP-based model to rapidly adapt to changing workload conditions and network dynamics, resulting in optimal resource allocation and improved system performance in cloud networks [30].

Algorithm 1: Algorithm for long-term potentiation (LTP) model for WorkLoad Scheduling in Cloud network

Initialize: Historical workload data D, Synaptic weights W, Learning rate μ

For each workload sampled_k In D, do

Convert $d_k$to feature vector $v_k$with contextual information

Adjust synaptic weights based on the Hebbian learning rule:

For each pair of features $v_j$and v_k do

$$\Delta W_{jk} = \mu \times v_j \times v_k$$

End for

For each new task t_{new} do

Calculate compatibility scores between $t_{new}$and resources

For each resource k, do

$$Compatibility\ (t_{new}, k) = v_{new}.\ w_k$$

End for End for

Allocate t_{new} to the resource with the highest compatibility score

Receive feedback on task completion times t_{comp}

Update synaptic weights

For each pair of features $v_j$and v_k do

$$\Delta Wjk = \mu \times (tcomp - tpred) \times vj \times vk$$

Evaluate performance metrics

End for until convergence or the desired number of iterations

3.2. Dispersive Fly Optimisation for Workload balance

The Dispersive Fly Optimization (DFO) handles workload balance using swarm intelligence techniques. The swarm intelligence technique-based resource optimisation is used to improve the performance of imbalanced data [6]. DFO is performed to solve continuous and discrete optimisation problems in a cloud environment and enhance and handle workload balance. The research explores the use of flies' swarming behaviour and diversity in the search space to conduct cost-sensitive learning on a specific set of datasets. Dispersive Fly Optimization (DFO) aims to analyse how the performance of imbalanced data can effectively address the challenges posed by diverse datasets and their associated costs, with the potential to yield valuable insights in various practical applications [31].

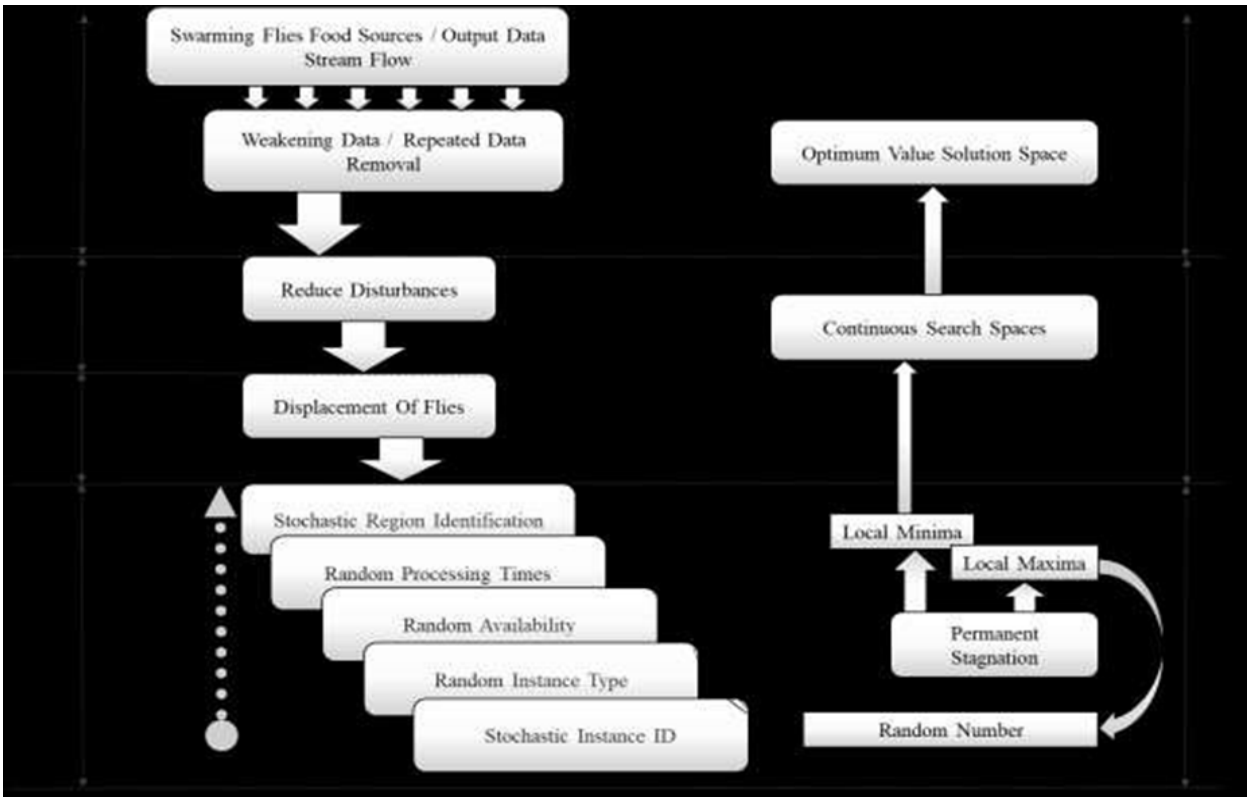

Figure 5. Dispersive Fly Optimization (DFO) for performance analysis of imbalanced data

Figure 5 illustrates the Dispersive Fly Optimization (DFO) analysis to improve the performance of imbalanced data-optimised Workload Scheduling in the Cloud Paradigm. The behaviour of swarms is influenced by swarming flies' food sources from the threat and convergence of output data stream flow, including conditions, attacks in search of booty presence, and the availability of data sources.

The external threats also make potential changes in the swarm convergence for optimal data source balancing. It will cause the potential disruption of swarm convergence. Swarm convergence might vary depending on the external threats, with similar convergence occurring in the coordinated [19]. This research algorithm considers the potential breaking or weakening of swarms after they have converged around the food source marker, thus accounting for the dynamic nature of flies' behaviour. Understanding swarm formation and dispersion dynamics is crucial for developing effective fly control and management strategies. It allows for the identification of vulnerabilities and the implementation of targeted interventions [7].

3.2.1. Dispersive Fly Optimization (DFO) for Using Swarm Intelligence

Dispersive Fly Optimization (DFO) is an algorithm inspired by the swarming behaviour of flies, which can be effectively applied to handle workload balancing in computational environments. DFO distributes tasks evenly across numerous processors or resources, emulating flies decentralised and self-organised nature, assuring optimal performance and efficiency [22]. In the context of workload balancing, each fly in the DFO algorithm is a potential solution to the problem. A fly's position represents a precise allocation of duties to resources. First, the fly population's position vectors are randomly assigned inside the provided constraints. Position vectors for a population of flies are defined as Equation 4:

$$p_k = (p_{i1}, p_{i2}, \ldots, p_{iT}); \; k = 1,2, \ldots, Pop_n \quad (4)$$

Where T is the number of tasks, Pop_n Is the number of flies (population size) and p_k is the assignment of task d to a resource for fly k. Each component, p_{i1}, is initialised randomly as Equation 5

$$pkd = pmin, d + R. (pmax, d - pmin, d), R\sim \cup (0,1) \quad (5)$$

Here, $p_{min,\ d}$ and $p_{max,\ d}$ are task's lower and upper bounds, respectively, and R is a random number taken from a uniform distribution U (0,1). After the population is established, the fitness of each fly is assessed using a fitness function that measures the quality of the workload distribution. For workload balancing, the fitness function often entails minimising the makespan (total execution time) and distributing the load across resources. A typical fitness function is defied in Equation 6:

$$Fitness(pk) = maxi\ (\sum d \epsilon Task\ assigned\ to\ i\ ET\ (d, i)) \quad (6)$$

This function computes the maximum load on resource i, where ET(d, i) represents the time required to run task d on resource i. Each fly's position is then adjusted following the best surrounding fly and the best fly in the entire swarm. This is similar to the social behaviour of flies that swarm towards promising regions. The position update equation is provided by Equation 7:

$$p_t{}^d = p_{t-1}^{nb,d} + \cup\ (0,1).\ (p_{t-1}^{sb,d} - p_{t-1}^{d}) \quad (7)$$

Here, ${}^{p_t nb_{-1}, d}$is the d-th component of the best neighbouring fly at time step t-1, and ${}^{p_t sb_{-1}, d}$ is the d-th component of the best fly in the whole swarm at time step −1. U(0,1) refers to a random number from a uniform distribution. To prevent stagnation and enhance search space exploration, stochastic disruption is introduced. This prevents the flies from becoming stuck in local optima [32]. Position vector components are reset with a probability determined by the disturbance threshold dt. The DFO method cycles through these phases until a stopping requirement is reached, such as a maximum number of function evaluations (e.g., 300,000) or a suitable fitness level. The algorithm is as follows: initialise the position vectors, evaluate the fitness of each fly, update the positions using the best nearby and swarm flies, apply stochastic disturbance, and iterate until convergence [8][41]. DFO efficiently distributes workload among resources, ensuring that no single resource is overwhelmed while minimising total execution time. Swarm intelligence techniques, characterised by the flies' collective behaviour and interactions, enable DFO to dynamically adapt to different workload scenarios and efficiently identify optimal or near-optimal solutions. This makes DFO a valuable tool for addressing workload balance difficulties in various computational environments [18][44].

Algorithm 2: Algorithm for Dispersive Fly Optimization (DFO) for Workload Balancing

Input: Pop_n-Number of flies (population size), T-Number of tasks, p_{min} and p_{max}-task assignments, dt: Disturbance threshold

Output: Best solution (task-resource allocation) for each fly-in Pop_ndo

for each task in Tdo

$$pkd = pmin, d + R.\ (pmax, d - pmin, d)$$

end for end for Set: Fitness = 0

while fitness < $Fitness_{max}$do Evaluate the fitness of each fly: for i = 1 to Pop_n do

$$Fitness(p_k) = max_i\ (\sum ET(d, i))\ d \epsilon Task\ assigned\ to\ i$$

end for

Fitness = Fitness + Pop_n

for each fly in Pop_ndo for each task in Tdo

$$p_t d = p_{t-1}^{nb,d} + \cup\ (0,1).\ (p_{t-1}^{sb,d} - p_{t-1}^{d})$$

if rand (0, 1) < dt, then

$$p_k{}^d = p_{min},\ d + R\ (0,1) * (p_{max},\ d - p_{min},\ d)$$

end if end for end for end while

Return the best solution found

The method involves two primary components: dynamic rule for positions of flies and best-found fly results for other flies within the swarm. The swarm can experience disturbances due to various reasons. This interruption can lead to the displacement reaction of flies, potentially leading to the breakthrough of better positions. A stochastic element (Stochastic Region Identification, Random Processing Times, Random Availability, Random Instance Type, Stochastic Instance ID) is introduced into the Random Number iteration process to account for this probabilistic behaviour [9]. Specifically, suppose a random number (RN) generated from a uniform distribution on the Permanent Stagnation PS (0,1) is less than a predefined disturbance threshold (Th). Individual components of the flies' position vectors are reset [33]. This stochastic element (Identification, Processing Times, Availability, Instance Type, Instance ID) allows for the potential impact of interruption on the flies' positions to be considered. The statement suggests that there will be a disruption to the ongoing stagnation, specifically around a probable local minimum (lm).

Dispersive Fly Optimization (DFO) perform as expected when applied as a population-based stochastic element, initially designed to explore and identify the probable local minimum (lm) value within the feasible iteration. The findings highlight DFO's superior performance across measures like Initialization, Fitness Evaluation, Dispersion, Social Interaction, Update Positions, Convergence Check, and Iteration [34]. A detailed analysis investigated the algorithm's diversity throughout the workload balance process. It also provided valuable insights into its robustness and effectiveness in solving complex, practical balance exploration and exploitation of complex workload scheduling with workload balance in cloud paradigm problems [35].

3.3. Transcranial random noise stimulation (tRNS)

Transcranial random noise stimulation (tRNS) explains its potential impact on Scheduling efficiency. tRNS could be attributed to the following: "Changes in nerve cell activity can happen when neurons are switched on or off, or when neuronal networks become more responsive to modulation" [36]. Scheduling efficiency is achieved by the output and input data stream flow, which can happen when cloud data in resource-level provisioning resource allocation or load balancing becomes more responsive to workload scheduling. Changes in the virtual machine i/o interception and Virtual Machine (VM) integration within the cloud environment and their resource utilisation to workload scheduling can potentially be influenced by transcranial Random Noise Stimulation (tRNS). This influence may extend to the workload scheduling efficiency, ultimately leading to alterations in the overall efficiency of the cloud network in a direction alternative to workload scheduling [10][43].

3.3.1. Transcranial Random Noise Stimulation (tRNS) for Scheduling efficiency

Transcranial random noise stimulation (tRNS) is a cutting-edge method that has garnered popularity for improving cognitive functions such as time management and scheduling efficiency. Applying random electrical noise currents to specific scalp regions, tRNS alters neuronal activity, altering cognitive functions such as time perception, decision-making, and task scheduling. In this talk, we will look at the fundamentals of tRNS and how it can be used to manage time and schedule efficiently, as well as include vital equations to help understand its mechanism and effectiveness [21]. The ability of tRNS to alter neuronal excitability via random noise currents is central to its premise.

The timing and scheduling of tRNS are critical in maximising its benefits on cognitive skills such as time management and scheduling efficiency [37]. During experimental sessions, tRNS is delivered sporadically and synced with task blocks to prevent adaptation to the stimulation. The duration of each task block T_{block} is calculated by adding the task time T_{task} and the rampup time $T_{ramp-up}$. This relationship can be stated mathematically as Equation 8 follows:

$$Tblock = Ttask + Tramp{-}up \quad (8)$$

Where, T_{task}represents the time allocated for performing the task and $T_{ramp-up}$Signifies the duration of the ramp-up phase before the task begins. We can maximise the effects of tRNS on cognitive processes by altering the time of its delivery inside each task block. tRNS's success in improving time management and scheduling efficiency stems from its capacity to alter neuronal activity in specific brain regions. According to research, tRNS can impact cortical excitability and synaptic plasticity, improving cognitive skills like attention, memory, and executive control. Targeting brain regions involved in time perception and decision-making, such as the prefrontal cortex, tRNS, can improve the brain's ability to organise, prioritise, and manage tasks more efficiently [11] [1].

Algorithm 3: Algorithm for Transcranial Random Noise Stimulation (tRNS) for Managing Time and Scheduling Efficiency

Input: I-Current intensity (mA), A-Electrode area (cm²), Z_threshold: Electrode impedance threshold (kΩ), T_{task}: Duration of each task block (seconds), Ramp_up_duration: Duration of the ramp-up phase (seconds), T_{block}: Total number of task blocks, T_{time}: Timing of task blocks

Output: Enhanced time management and scheduling efficiency

Calculate current density J = I / A Initialize: T_{time}, stimulation timing

for each T_{block}in $T_{sc}hedule$do

Start tRNS stimulation

while $Ttask <$ $Tduration$do

Monitor electrode impedance:

if impedance > $Z_threshold$then

Adjust stimulation parameters end if continue tRNS stimulation end while End tRNS stimulation end for Furthermore, the random nature of the noise currents used during tRNS may aid in breaking down fixed patterns of brain activity, promoting flexibility and adaptation in task management. This stochastic resonance phenomenon increases the brain's sensitivity to weak signals, allowing for more efficient temporal information processing and better decision-making under time restrictions [20][42]. Transcranial random noise stimulation (tRNS) is a promising way to improve time management and scheduling efficiency by modifying neuronal activity in critical brain regions. Researchers can use precise timing and scheduling methods to enhance the effects of tRNS on cognitive processes such as time perception, decision-making, and task scheduling. Understanding the underlying mechanisms and using proper stimulation levels might help improve cognitive performance activities that require efficient time management and scheduling [38].

VI. RESULT AND DISCUSSION

The proposed Dispersive Fly Optimization (DFO) was experimentally evaluated utilizing a simulation-based setup constructed in Matlab R2018b. We compared DFO to three common metaheuristic algorithms: Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and the traditional Whale Optimization Algorithm (WOA). The primary evaluation metrics included overall cost, load cost, throughput, memory demand measured across 100 to 1000 jobs, and the number of virtual machines in cloud computing scenarios. We employed 50 search agents and 40 virtual machines (VMs), with job and VM parameters adjusted at random within preset ranges.

Total Cost: Represents the total cost of performing tasks in the cloud environment. It was calculated as the total of task execution and VM deployment expenses.

$$Total\ Cost = Task\ Execution\ Cost + VM\ Deployment\ Cost$$

Load Cost: Measures the balance of workload allocation across VMs. Lower load cost indicates better workload balancing.

$$Load\ Cost = \frac{\sum_{k=1}^{N}(Task\ Load_k - Average\ Load)_2}{N}$$

Throughput: It indicates the rate at which tasks are completed in the cloud. High throughput indicates efficient resource utilization.

$$Throughput = \frac{Total\ Tasks\ Completed}{Total\ Time}$$

Memory Demand: Reflects the amount of memory used by the cloud infrastructure. Supports resource provisioning and management.

$$Memory\ Demand = \sum_{k=1}^{N} Task\ Memory\ Requirement_k$$

Task Completion Time (TCT): Measures the average time required to accomplish tasks in the cloud environment.

$$TCT = \frac{\sum_{i=1}^{N} T_i}{N}$$

Where N is the total number of tasks. T_i is the completion time of the \(i^{th} \) task.

Resource Utilization Rate (RUR): The proportion of available resources that are actively used to complete tasks.

$$RUR = \frac{Total\ Resources\ Used}{Total\ Available\ Resources} \times 100\%$$

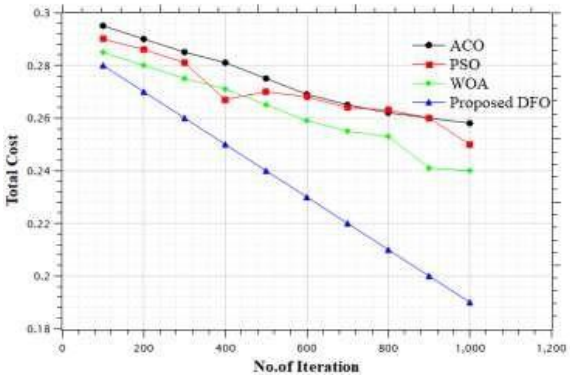

Figure 6: Overall Cost Incurred by Various Optimization Strategies Over Iterations

Figure 6 depicts the overall cost of various optimization strategies over several iterations. At the first iteration of 100, ACO has the most significant total cost of 0.295, followed by PSO (0.290), WOA (0.285), and the Proposed DFO (0.280). As iterations progress, the total cost of all algorithms reduces. However, the pace of reduction differs between them. At 1000 iterations, the Proposed DFO achieves the most efficient cost reduction, with a total cost of 0.190,

followed by WOA (0.240), PSO (0.250), and ACO (0.258). Overall, the proposed DFO consistently outperforms ACO, PSO, and WOA in terms of lowering overall cost over iterations.

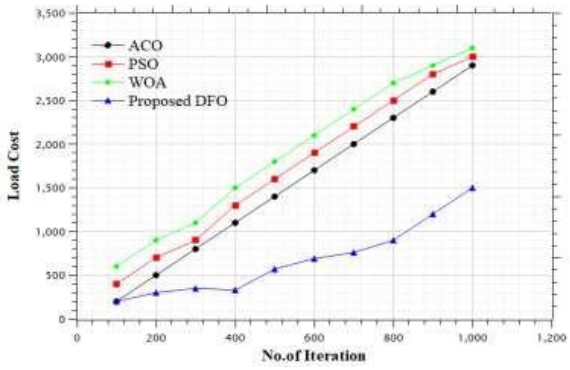

Figure 7: Load Costs Associated with Various Optimization Strategies

Figure 7 shows the load costs associated with various optimization strategies. At the first iteration, ACO has the lowest load cost of 200, while PSO, WOA, and the Proposed DFO have higher load costs of 400, 600, and 200. As iterations progress, load costs for all algorithms rise. However, the pace of increase varies between them. For example, ACO shows a consistent increase in load cost over iterations, reaching 2900 after 1000 iterations, whereas PSO and WOA exhibit a similar rising trend, with load costs peaking at 3000 and 3100, respectively. In comparison, the proposed DFO shows a more progressive increase in load cost, reaching 1500 after 1000 iterations. Overall, the Proposed DFO has a lower load cost than ACO, PSO, and WOA across all iterations, making it a more efficient load management option.

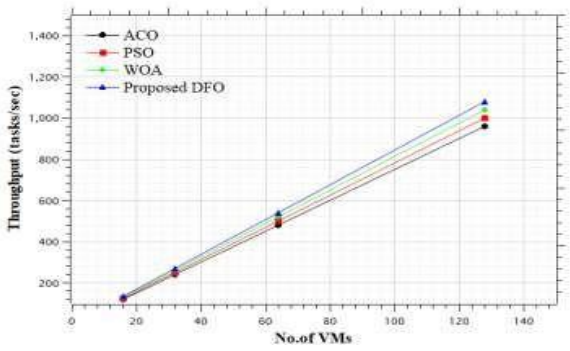

Figure 8: Throughput (Tasks/sec) for Various Optimization Techniques with Different Numbers of VMs

Figure 8 depicts the throughput, measured in jobs per second, for several optimization techniques, including the proposed Dispersive Fly Optimization (DFO). For all algorithms, throughput figures progressively rise as the number of VMs increases. For example, with 16 VMs, ACO achieves 120 tasks/sec, whereas DFO delivers 135 tasks/sec, representing a 12% improvement over ACO. Similarly, with 128 VMs, ACO achieves 960 tasks/sec, whereas DFO obtains 1080 tasks/sec, representing a 12.5% increase above ACO. When comparing algorithms, DFO consistently outperforms ACO, PSO, and WOA across all VM configurations. For example, with 64 VMs, DFO achieves 540 tasks/sec, whereas the closest competitor, WOA, achieves 520 tasks/sec, indicating a 3.8% improvement with DFO. This trend is consistent across all VM configurations, illustrating DFO's outstanding performance in optimizing throughput and improving task execution efficiency in cloud environments.

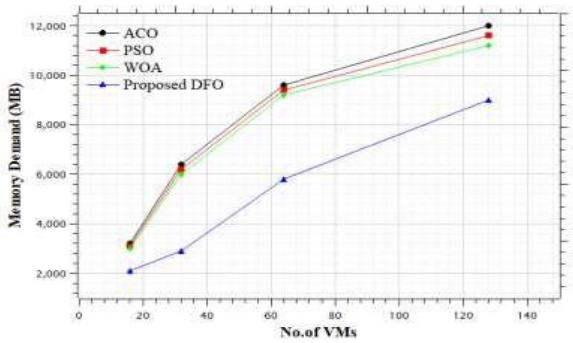

Figure 9: Memory Demand (MB) Across Different Numbers of VMs Using Various

Optimization Strategies

Figure 9 depicts memory demand in megabytes (MB) across different numbers of virtual machines (VMs) using various optimization strategies. Memory requirements for all algorithms increase equally as the number of virtual machines (VM) increases. For example, with 16 VMs, ACO requires 3200 MB while DFO uses 2100 MB, resulting in a 34.4% savings with DFO. DFO consistently has lower memory use than ACO, PSO, and WOA across all VM configurations. With 64 VMs, DFO uses 5800 MB, whereas WOA uses 9200 MB, resulting in a significant 37.5% reduction. This pattern emphasizes DFO's efficiency in-memory optimization, showcasing its ability to reduce resource waste and improve system performance in cloud environments.

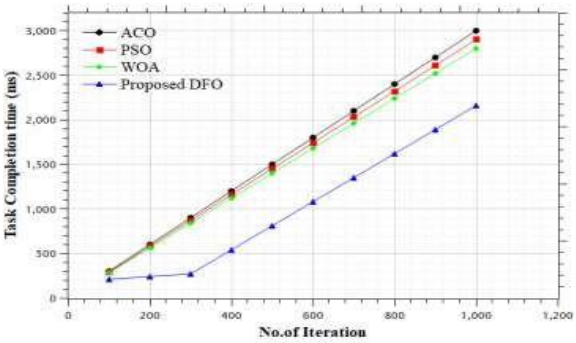

Figure 10: Task Completion Time (ms) for Various Optimization Techniques Over Iterations

Figure 10 shows the millisecond task completion time (TCT) for four optimization techniques with varying iteration counts. At 100 iterations, the proposed DFO has a TCT of 210 ms, which is much lower than ACO's 300 ms, PSO's 290 ms, and WOA's 280 ms. This shows DFO's higher efficiency early in the optimization process. This tendency persists as the number of iterations increases, with DFO having the lowest TCT across all counts. For example, at 500 iterations, DFO has a TCT of 810 ms, compared to ACO's 1500 ms, PSO's 1450 ms, and WOA's 1400 ms, indicating a significant decrease. At 1000 iterations, the suggested DFO obtains a TCT of 2160 ms, while ACO, PSO, and WOA record 3000 ms, 2900 ms, and 2800 ms, respectively. This constant result demonstrates DFO's effectiveness in reducing task completion time, outperforming other algorithms across all iterations. DFO's capacity to attain reduced TCT values illustrates its efficacy in job scheduling and resource allocation.

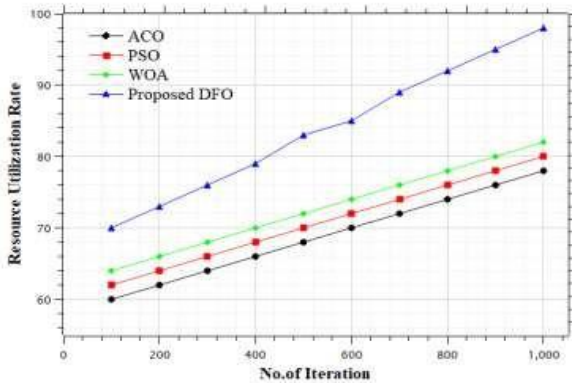

Figure 11: Resource Utilization Rate for Various Optimization Techniques Over Iteration

Figure 11 displays the Resource Utilization Rate (RUR) for four optimization techniques with varying iteration counts. RUR consistently increases with the number of iterations for all methods, demonstrating that resource consumption efficiency improves over time. For example, after 100 cycles, ACO obtains an RUR of 60%, whereas DFO achieves 70%, representing a 10% improvement over ACO. RUR comparisons among algorithms indicate interesting trends. With 400 cycles, DFO gets a RUR of 79%, while its nearest competitor, WOA, earns 70%, suggesting a significant 9% gain in resource use efficiency with DFO. Across all iteration counts, the suggested DFO consistently outperformed ACO, PSO, and WOA regarding RUR. For example, with 800 iterations, DFO gets an RUR of 92%, but ACO only achieves 74%, indicating a significant 18% increase in resource use efficiency with DFO. This graph demonstrates DFO's ability to optimize resource allocation and improve system performance during optimisation scenarios.

V. CONCLUSION

A research article explored a four-stage task-scheduling framework within a cloud paradigm. The framework involved task scheduling and performance processes and proposed a schedulability analysis model for dataset-based static workloads. The model operates under the 'long-term potentiation (LTP) model' for workload scheduling. Additionally, the article introduced a new workload scheduling policy designed to optimize workload balance using dispersive fly optimization based on swarm intelligence techniques. The research focused on four policy variants: Stage 1: Input Data Stream Flow, Stage 2: Virtual Machine I/O Interception (synapse), Stage 3: Virtual Machine (VM) Integration (Patterns of stimulation), and Stage 4: Output Data Stream Flow. Each stage is executed with policies based on resourcelevel provisioning and resource allocation. In addition, a novel approach called transcranial random noise stimulation (tRNS) was introduced and designed to improve time management and scheduling efficiency. It also explores the potential integration of this method with other techniques. Additionally, the paper showcases how the proposed analysis can be used offline to ensure the scheduling of static task sets with mode changes. Further research will involve

implementing candidate algorithms, such as genetic and linear programming, in a natural cloud computing system. The future focus will be enhancing the scheduling method by minimizing energy consumption while maintaining service quality. Moreover, future articles aim to expand the new carrying estimation technology to accommodate various task models, such as selfsuspension tasks.

REFERENCE

[1] A. Alimolu and C. Özturan, "An autonomous blockchain-based workflow execution broker for e-science," Cluster Computing, 2024. doi: 10.1007/s10586-024-04534-z.

[2] A. Awan, M. Aleem, A. Hussain et al., "DFARM: A deadline-aware fault-tolerant scheduler for cloud computing," Cluster Computing, 2024. doi: 10.1007/s10586-024-04419-1.

[3] B. Kruekaew and W. Kimpan, "Multi-objective task scheduling optimization for load balancing in cloud computing environment using hybrid artificial bee colony algorithm with reinforcement learning," IEEE Access, vol. 10, pp. 17803–17818, 2022, doi: 10.1109/ACCESS.2022.3149955.

[4] S. Bansal and H. Aggarwal, "A multiobjective optimisation of task workflow scheduling using hybridisation of PSO and WOA algorithms in cloud-fog computing," Cluster Computing, 2024. doi: 10.1007/s10586-024-04522-3.

[5] C.-C. Hsu, S. Shrivastava, S. Pratik, S. Chandrasekaran, and T.-Y. Tseng, "ZTO/MgO-based optoelectronic synaptic memristor for neuromorphic computing," IEEE Transactions on Electron Devices, vol. 70, no. 3, pp. 1048–1054, Mar. 2023, doi: 10.1109/TED.2023.3237666.

[6] C. Wang, P. Yang, J. Hou, Z. Liu, and N. Zhang, "Dependence-aware multi-task scheduling for edge video analytics with accuracy guarantee," IEEE Internet of Things Journal, 2024, doi: 10.1109/JIOT.2024.3397296.

[7] A. Chakraborty, M. Kumar, and N. Chaurasia, "An intelligent offloading and resource allocation using a fuzzy-based HHGA algorithm for IoT applications," Cluster Computing, 2024. doi: 10.1007/s10586-024-04536-x.

[8] S. K. Chowdhary and A. L. N. Rao, "QoS and reliability aware matched bald eagle task scheduling framework based on IoT-cloud in educational applications," Cluster Computing, 2024. doi: 10.1007/s10586-024-04415-5.

[9] D. Casini, A. Biondi, and G. Buttazzo, "Handling transients of dynamic real-time workload under EDF scheduling," IEEE Transactions on Computers, vol. 68, no. 6, pp. 820–835, Jun. 2019, doi: 10.1109/TC.2018.2882451.

[10] D. Casini, A. Biondi, and G. Buttazzo, "Task splitting and load balancing of dynamic real-time workloads for semi-partitioned EDF," IEEE Transactions on Computers, vol. 70, no. 12, pp. 2168–2181, Dec. 2021, doi: 10.1109/TC.2020.3038286.

[11] F. Zhang, J. Cao, W. Tan, S. U. Khan, K. Li, and A. Y. Zomaya, "Evolutionary scheduling of dynamic multitasking workloads for big-data analytics in elastic cloud," IEEE Transactions on Emerging Topics in Computing, vol. 2, no. 3, pp. 338–351, Sept. 2014, doi: 10.1109/TETC.2014.2348196.

[12] G. Ye, F. Gao, J. Fang, and Q. Zhang, "Joint workload scheduling in geo-distributed data centers considering UPS power losses," IEEE Transactions on Industry Applications, vol. 59, no. 1, pp. 612–626, Jan.–Feb. 2023, doi: 10.1109/TIA.2022.3214186.

[13] R. Ghafari and N. Mansouri, "A novel energy-based task scheduling in fog computing environment: An improved artificial rabbits optimisation approach," Cluster Computing, 2024. doi: 10.1007/s10586-024-04396-5.

[14] H. Yuan, J. Bi, W. Tan, and B. H. Li, "CAWSAC: Cost-aware workload scheduling and admission control for distributed cloud data centers," IEEE Transactions on Automation Science and Engineering, vol. 13, no. 2, pp. 976–985, Apr. 2016, doi: 10.1109/TASE.2015.2427234.

[15] J. Meng et al., "Li-ion doped artificial synaptic memristor for highly linear neuromorphic computing," IEEE Electron Device Letters, vol. 43, no. 12, pp. 2069–2072, Dec. 2022, doi: 10.1109/LED.2022.3211520.

[16] J. Song et al., "Photoelectric synaptic device based on bilayered OR/OP-InGaZnO for neuromorphic computing," IEEE Electron Device Letters, vol. 45, no. 1, pp. 120–123, Jan. 2024, doi: 10.1109/LED.2023.3335628.

[17] Z. Jafari, A. Habibizad Navin, and A. Zamanifar, "Task scheduling approach in fog and cloud

computing using Jellyfish Search (JS) optimiser and improved Harris Hawks optimisation (IHHO) algorithm enhanced by deep learning," Cluster Computing, 2024. doi: 10.1007/s10586-024-04347-0.

[18] K. Kamal, A. Thakur, and J. Singh, "Emulating switching from short-term to long-term plasticity of bio-synapse using split gate MOSFET," IEEE Transactions on Nanotechnology, vol. 21, pp. 449–454, 2022, doi: 10.1109/TNANO.2022.3198223.

[19] K. Zhang et al., "TENSILE: A tensor granularity dynamic GPU memory scheduling method toward multiple dynamic workloads system," IEEE Transactions on Knowledge and Data Engineering, vol. 35, no. 8, pp. 8630–8643, Aug. 2023, doi: 10.1109/TKDE.2022.3221426.

[20] L. Cheng, Y. Wang, F. Cheng, C. Liu, Z. Zhao, and Y. Wang, "A deep reinforcement learning-based preemptive approach for cost-aware cloud job scheduling," IEEE Transactions on Sustainable Computing, 2023, doi: 10.1109/TSUSC.2023.3303898.

[21] L. Tian, Q. Liu, L. Song, Z. Jin, L. Dong, and Y. Zheng, "Design and optimisation of resonance-based wireless power transmission for millimeter-sized planar square inductors micro-magnetic stimulation," IEEE Transactions on Instrumentation and Measurement, vol. 72, pp. 1–10, 2023, Art. no. 4009010, doi: 10.1109/TIM.2023.3292965.

[22] L. Wan, W. Zheng, and X. Yuan, "Efficient inter-device task scheduling schemes for multi-device co-processing of data-parallel kernels on heterogeneous systems," IEEE Access, vol. 9, pp. 59968–59978, 2021, doi: 10.1109/ACCESS.2021.3073955.

[23] L. Zeng, Y. Wang, X. Fan, and C. Xu, "Raccoon: A novel network I/O allocation framework for workload-aware VM scheduling in virtual environments," IEEE Transactions on Parallel and Distributed Systems, vol. 28, no. 9, pp. 2651–2662, Sept. 2017, doi: 10.1109/TPDS.2017.2685386.

[24] H. Lin, G. Liu, W. Lin et al., "A novel virtual machine consolidation algorithm with server power mode management for energy-efficient cloud data centers," Cluster Computing, 2024. doi: 10.1007/s10586-024-04555-8.

[25] J. Liu, Y. Ding, and Y. Liu, "A balanced leader election algorithm based on replica distribution in Kubernetes cluster," Cluster Computing, 2024. doi: 10.1007/s10586-024-04333-6.

[26] M. D'Amico and J. C. Gonzalez, "Energy hardware and workload aware job scheduling towards interconnected HPC environments," IEEE Transactions on Parallel and Distributed Systems, 2021, doi: 10.1109/TPDS.2021.3090334.

[27] P. Zhang and M. Zhou, "Dynamic cloud task scheduling based on a two-stage strategy," IEEE Transactions on Automation Science and Engineering, vol. 15, no. 2, pp. 772–783, Apr. 2018, doi: 10.1109/TASE.2017.2693688.

[28] Q. Li et al., "Organic optoelectronic synaptic devices for energy-efficient neuromorphic computing," IEEE Electron Device Letters, vol. 43, no. 7, pp. 1089–1092, Jul. 2022, doi: 10.1109/LED.2022.3180346.

[29] Q. Zhou, G. Li, and J. Li, "Improved carry-in workload estimation for global multiprocessor scheduling," IEEE Transactions on Parallel and Distributed Systems, vol. 28, no. 9, pp. 2527–2538, Sept. 2017, doi: 10.1109/TPDS.2017.2679195.

[30] S. A. Khan, M. Oli-Uz-Zaman, and J. Wang, "PAWN: Programmed analog weights for non-linearity optimization in memristor-based neuromorphic computing system," IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 13, no. 1, pp. 436–444, Mar. 2023, doi: 10.1109/JETCAS.2023.3235658.

[31] S. Nabi, M. Ibrahim, and J. M. Jimenez, "DRALBA: Dynamic and resource aware load balanced scheduling approach for cloud computing," IEEE Access, vol. 9, pp. 61283–61297, 2021, doi: 10.1109/ACCESS.2021.3074145.

[32] D. Shahmirzadi, N. Khaledian, and A. M. Rahmani, "Analysing the impact of various parameters on job scheduling in the Google cluster dataset," Cluster Computing, 2024. doi: 10.1007/s10586-024-04377-8.

[33] Z. Shamsa, A. Rezaee, S. Adabi et al., "A distributed load balancing method for IoT/fog/cloud environments with volatile resource support," Cluster Computing, 2024. doi: 10.1007/s10586-024-04403-9.

[34] A. Souri, S. E. Mood, M. Gao et al., "Tournament based equilibrium optimisation for minimising energy consumption on dynamic task scheduling in cloud-edge computing," Cluster Computing, 2024. doi: 10.1007/s10586-024-04489-1.

[35] N. Thapliyal and P. Dimri, "Task scheduling using fuzzy logic with best-fit-decreasing for cloud computing environment," Cluster Computing, 2024. doi: 10.1007/s10586-024-04378-7.

[36] A. Ullah, Z. Alomari, S. Alkhushayni et al., "Improvement in task allocation for VM and reduction of makespan in IaaS model for cloud computing," Cluster Computing, 2024. doi: 10.1007/s10586-024-04539-8.

[37] X. Li, P. Garraghan, X. Jiang, Z. Wu, and J. Xu, "Holistic virtual machine scheduling in cloud datacenters towards minimizing total energy," IEEE Transactions on Parallel and Distributed Systems, vol. 29, no. 6, pp. 1317–1331, Jun. 2018, doi: 10.1109/TPDS.2017.2688445.

[38] X. Wang, Y. Jin, W. Du, and J. Wang, "Evolving dual-threshold Bienenstock-Cooper-Munro learning rules in echo state networks," IEEE Transactions on Neural Networks and Learning Systems, vol. 35, no. 2, pp. 1572–1583, Feb. 2024, doi: 10.1109/TNNLS.2022.3184004.

[39] X. Xu, L. Cao, and X. Wang, "Adaptive task scheduling strategy based on dynamic workload adjustment for heterogeneous Hadoop clusters," IEEE Systems Journal, vol. 10, no. 2, pp. 471–482, Jun. 2016, doi: 10.1109/JSYST.2014.2323112.

[40] X. Zhu, K. M. Sim, J. Jiang, J. Wang, C. Chen, and Z. Liu, "Agent-based dynamic scheduling for earth-observing tasks on multiple airships in emergency," IEEE Systems Journal, vol. 10, no. 2, pp. 661–672, Jun. 2016, doi: 10.1109/JSYST.2014.2327069.

[41] Y. Chen et al., "Stochastic workload scheduling for uncoordinated datacenter clouds with multiple QoS constraints," IEEE Transactions on Cloud Computing, vol. 8, no. 4, pp. 1284–1295, Oct.–Dec. 2020, doi: 10.1109/TCC.2016.2586048.

[42] Y. Fang et al., "Two-terminal photoelectric dual modulation synaptic devices for face recognition," IEEE Electron Device Letters, vol. 44, no. 2, pp. 241–244, Feb. 2023, doi: 10.1109/LED.2022.3228944.

[43] Y. Zhu et al., "Photoelectric synapse based on InGaZnO nanofibers for high precision neuromorphic computing," IEEE Electron Device Letters, vol. 43, no. 4, pp. 651–654, Apr. 2022, doi: 10.1109/LED.2022.3149900.

[44] Z. Zhang, F. Zhang, Z. Xiong, K. Zhang, and D. Chen, "LsiA3CS: Deep reinforcement learning-based cloud-edge collaborative task scheduling in large-scale IIoT," IEEE Internet of Things Journal, 2024, doi: 10.1109/JIOT.2024.3386888.