I.J. Information Engineering and Electronic Business, **2026, 3, 142-162**
Published Online on June 8, 2026 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijieeb.2026.03.09

AI-Based Metaheuristic Algorithm for Reducing Cost and VM Failure Rate in Task Scheduling within Cloud Computing Environments

Santhosh Kumar Medishetti*
Nalla Narasimha Reddy Education Society's Group of Institutions, Hyderabad, Telangana, 500088, India
E-mail: medishettysantosh@gmail.com
ORCID iD: https://orcid.org/0000-0001-5936-6582
*Corresponding author

G. Soma Sekhar
Geethanjali College of Engineering and Technology, Hyderabad, Telangana, 501301, India
E-mail: somasekharonline@yahoo.in
ORCID iD: https://orcid.org/0000-0002-9265-0929

Kommuri Venkatrao
Lakireddy Bali Reddy College of Engineering (Autonomous), Mylavaram, Andra Pradesh, 521230, India
E-mail: venkatkvr789@gmail.com
ORCID iD: https://orcid.org/0009-0001-1954-3748

Rani Sailaja Velamakanni
Nalla Narasimha Reddy Education Society's Group of Institutions, Hyderabad, Telangana, 500088, India
E-mail: vranisailaja@gmail.com
ORCID iD: https://orcid.org/0009-0001-7806-5483

Received: October 4, 2025; Revised: December 21, 2025; Accepted: February 2, 2026; Published: June 8, 2026

Abstract: Scheduling is an NP-hard problem, and heuristic algorithms are unable to find approximate solutions within a feasible time frame. In Cloud Computing (CC) environments, efficient Task Scheduling (TS) plays a critical role in minimizing operational expenses and enhancing system reliability. This paper presents a novel task scheduling approach that uses the Coati Optimization Algorithm (COA) to address two pivotal challenges: reducing the total cost (sum of computational cost and communication cost) and minimizing Virtual Machine (VM) failure rates. Inspired by the cooperative foraging and adaptive behavior of coatis in dynamic environments, the proposed algorithm leverages intelligent exploration and exploitation strategies to identify optimal task-to-VM mappings under fluctuating workloads. The COA incorporates cost-awareness and failure probability metrics into its fitness function to ensure robust scheduling decisions that align with budgetary constraints and fault tolerance requirements. To assess the performance of the proposed model, comprehensive simulations were conducted using the CEA-Curie real-world workload. The results were compared against three state-of-the-art approaches, MoHHOTS, RTATSA2C, and TS-GWO. Experimental evaluations demonstrate that COA significantly outperforms these existing methods by achieving a 19.8% reduction in overall cost and a 22.5% decrease in VM failure rate. These findings demonstrate that COA offer a promising pathway toward sustainable, cost-effective, and resilient task execution in large-scale cloud infrastructures, particularly under diverse and realistic workload scenarios.

Index Terms: NP-hard, Cloud Computing, Task Scheduling, Coati Optimization, Foraging, Cost, VM Failure Rate

1. Introduction

Cloud Computing (CC) has emerged as a dominant paradigm in modern computing, offering scalable, flexible, and cost-efficient access to computing resources through virtualized infrastructures. It enables users to deploy and manage applications without the burden of acquiring and maintaining physical hardware. One of the central operational

challenges in this environment is task scheduling the process of assigning user-submitted tasks to available Virtual Machines (VMs) [1,2]. Effective task scheduling ensures optimal resource utilization, minimizes service delays, and improves the overall throughput of cloud systems [3]. As cloud workloads become increasingly heterogeneous and dynamic, designing intelligent scheduling mechanisms becomes a vital component of cloud service delivery.

Efficient Task Scheduling (TS) directly influences key performance indicators such as execution time, makespan, resource utilization, energy consumption, and cost. Moreover, it ensures Quality of Service (QoS) compliance by guaranteeing that tasks meet their required deadlines and Service Level Agreements (SLAs) [4]. In large-scale cloud environments, the ability to schedule tasks in a timely and resource-aware manner determines the operational success of cloud service providers. Without efficient scheduling, systems suffer from increased latency, higher resource wastage, and poor user satisfaction. Therefore, TS has become a high-priority research area in CC [5].

Among various optimization objectives, minimizing the operational cost and ensuring system reliability are two of the most critical. The operational cost in cloud computing generally consists of computational cost, which is incurred for CPU usage and memory consumption, and communication cost, which is associated with the transfer of data between tasks and VMs. Simultaneously, maintaining a low VM failure rate is essential to guarantee fault-tolerant task execution. VM failures can lead to data loss, execution delays, and even SLA violations [6]. An ideal scheduling strategy must therefore balance economic constraints with high levels of reliability. Addressing both cost and VM failures simultaneously is a challenging multi-objective optimization problem. Cost minimization might inadvertently assign tasks to less reliable or overloaded VMs, increasing the likelihood of execution failure [7]. On the other hand, focusing exclusively on reliability may lead to inefficient use of resources and elevated costs. Moreover, the volatile and unpredictable nature of cloud workloads further complicates the scheduling process. Traditional scheduling approaches, such as static algorithms or rule-based heuristics, often lack the flexibility and intelligence to adapt to such dynamic conditions, resulting in sub-optimal task-to-VM mappings.

To overcome the limitations of traditional methods, researchers have turned to bio-inspired metaheuristic algorithms that mimic natural processes such as evolution, swarm intelligence, and animal foraging behaviors. These algorithms are well-suited for complex, high-dimensional optimization problems like cloud scheduling, as they provide robust mechanisms for global exploration and local exploitation. Techniques such as PSO, ACO, WOA, and GA have been widely used with varying degrees of success [8]. However, their performance can degrade in large-scale, real-time cloud environments, primarily due to slow convergence rates, premature convergence, and poor adaptability. In response to these limitations, this study introduces a novel metaheuristic algorithm known as the COA. This algorithm is inspired by the social foraging behavior of coatis mammals native to Central and South America known for their intelligence, cooperation, and adaptability [9]. Coatis forage in groups, where roles dynamically shift between leaders and followers based on environmental conditions and resource availability. By modelling this behavior computationally, COA provides a powerful mechanism for exploring the search space and converging on optimal solutions for task scheduling in a balanced and adaptive manner.

The core working principle of COA lies in its hybrid role-based strategy, where virtual coatis representing candidate solutions are divided into leaders and followers. Leaders guide the search based on previously discovered high-quality solutions, while followers explore surrounding areas to diversify the search process [10]. Periodic role-switching prevents the algorithm from stagnating in local optima and promotes global search capabilities. Additionally, COA introduces adaptive movement patterns and interaction rules that enhance convergence speed and accuracy. This dynamic balance between exploration and exploitation makes COA well-suited for scheduling tasks in volatile and complex cloud environments. A critical innovation in COA lies in its fitness function, which simultaneously evaluates solutions based on two core metrics: the total operational cost (sum of computational and communication costs) and the VM failure rate. By incorporating weighted coefficients, the fitness function can be adjusted to prioritize specific objectives based on the user's or system's needs. For instance, under high-budget constraints, more weight can be assigned to cost minimization, while for mission-critical applications, failure rate may be given higher importance. This customizable and multi-objective approach allows COA to be flexible and adaptable across different use cases.

The significance of this work lies in its dual-objective optimization focus and the application of a novel algorithm inspired by nature. Unlike many existing models that prioritize only a single objective or fail to incorporate real-world constraints like VM failures, COA introduces a holistic and adaptive approach. Its ability to simultaneously minimize cost and reduce failure rates makes it highly applicable to real-time, large-scale cloud environments where performance and reliability are equally important. Furthermore, the integration of social behavior into algorithm design represents a unique advancement in metaheuristic-based scheduling. The scope of this research is directed towards Infrastructure-as-a-Service (IaaS) cloud models, where users lease virtualized resources to deploy applications. In such settings, scheduling decisions must consider not only resource availability but also performance variability and potential failures. This study focuses on heterogeneous VM environments with dynamic workloads, making the proposed approach applicable to commercial cloud platforms like AWS, Azure, and Google Cloud. The algorithm's design is intentionally made modular and scalable, facilitating its deployment in distributed systems and hybrid cloud-fog architectures.

When compared with traditional and contemporary algorithms, COA presents multiple advantages. While classical heuristics may quickly converge, they often get trapped in local optima. Even advanced metaheuristics like PSO and GA show limitations in handling dual-objective optimization and workload volatility. COA, with its adaptive leader-

follower strategy and multi-objective fitness function, not only overcomes these limitations but also exhibits faster convergence, greater robustness, and better generalization across varied cloud environments. These characteristics make COA a superior alternative for cost- and reliability-aware scheduling. Despite significant progress in cloud task scheduling, a persistent research gap remains in developing models that jointly optimize both total cost and VM reliability using real-world workload data. Existing algorithms often treat failure rate as a secondary metric or neglect communication costs altogether. This leads to inefficient scheduling, resource overutilization, and task execution failures. Therefore, the problem addressed in this research is the absence of an intelligent, adaptive, and multi-objective scheduling algorithm that can handle real-time constraints while ensuring economic efficiency and operational reliability. This study is driven by three primary objectives:

1. To develop a Coati-inspired task scheduling algorithm that minimizes both computational and communication costs as well as VM failure rate.
2. To evaluate the performance of the proposed COA model against leading metaheuristic algorithms using a real-world cloud workload.
3. To demonstrate the scalability, flexibility, and reliability of COA under dynamic, heterogeneous cloud environments and varying QoS demands.

Although COA belongs to the broader class of population-based metaheuristics, it differs fundamentally from algorithms such as GWO and HHO in its behavioral modeling and decision dynamics. In COA, leadership is not fixed or rank-based; instead, individuals dynamically switch roles between leader and follower based on real-time fitness improvements and environmental feedback. This adaptive role-switching mechanism enhances responsiveness to workload volatility, which is critical in cloud environments. Furthermore, COA integrates workflow-awareness and VM failure sensitivity directly into the fitness evaluation, whereas existing algorithms typically treat reliability as a secondary or post-processing metric. These characteristics collectively establish COA as a distinct and problem-driven optimization framework rather than a variant of existing leader-follower algorithms. To rigorously evaluate the effectiveness of COA, the CEA-Curie workload dataset is used. This real-world workload captures job submissions from a large-scale high-performance computing environment, including metadata about task size, execution time, and failure characteristics. Its diversity and realism make it ideal for testing the robustness and applicability of cloud scheduling algorithms. This study benchmarks COA's performance using this dataset to ensure external validity and practical relevance of the results. In order to establish the effectiveness of COA, it is compared with three cutting-edge scheduling algorithms are, MoHHOTS known for its flexible objective balancing, RTATSA2C a reinforcement learning-based approach, and TS-GWO a hybrid metaheuristic combining memory and social behavior.

These algorithms represent the current state of the art in task scheduling, and comparing COA against them helps highlight its competitive advantages. The study uses two primary evaluation metrics: total cost and VM failure rate. Total cost is computed as the sum of CPU resource usage and data communication overhead. VM failure rate is measured by the ratio of task execution failures to total scheduled tasks. Secondary performance indicators include execution time, makespan, and throughput, which provide a comprehensive view of scheduling efficiency. The use of multiple metrics ensures that the assessment reflects both economic and operational aspects of cloud service delivery. The contributions of this research are, it introduces the COA, a novel metaheuristic approach based on natural social foraging behavior. It proposes a multi-objective fitness model that balances cost and reliability. The study also presents a thorough experimental analysis using a real-world workload and compares COA with recent high-performance scheduling algorithms. These contributions not only advance the field of cloud computing but also open avenues for biologically inspired computing in other complex optimization domains.

The rest of the paper is structured as follows, Section 2 provides an extensive review of related research in task scheduling and optimization in cloud environments. Section 3 details the architecture and operational mechanisms of the proposed Coati Optimization Algorithm. Section 4 outlines the experimental setup, including dataset description, parameter settings, and evaluation metrics also discusses the results, performance comparisons, and analytical insights. Lastly, Section 5 concludes the paper and provides future research directions. The below table 1 depicts the acronyms used in this study.

Table 1. Description of acronyms.

Notation	Description
TS	Task Scheduling
COA	Coati Optimization Algorithm
CC	Cloud Computing
SLAs	Service Level Agreements
VM	Virtual Machine
QoS	Quality of Service

2. Literature Survey

Ferdaus et al. [11] proposed an ACO based method for dynamic VM consolidation in an effort to reduce resource optimization and energy consumption. While effective in reducing energy usage, their method primarily focuses on resource consolidation rather than task scheduling, and it doesn't explicitly address VM failure rates. In contrast, our COA-based scheduler directly targets task scheduling with an emphasis on both cost efficiency and VM reliability. Saxena et al. [12] proposed a Whale Optimization Genetic Algorithm (WOGA) for secure and energy-efficient VM placement. Their framework emphasizes security and energy consumption but lacks consideration for communication costs and VM failure rates. Our COA approach extends beyond by incorporating communication costs and VM reliability into the scheduling process, providing a more holistic optimization. Tuli et al. [13] developed MetaNet, a surrogate model that dynamically selects cost-optimal schedulers in cloud environments. While MetaNet adapts to workload changes, it primarily focuses on execution cost without addressing VM failure rates. Our COA-based scheduler concurrently optimizes for both cost and reliability, ensuring robust performance even in the presence of VM failures. Abualigah et al. [14] introduced TS-GWO, leveraging the GWO for IoT task scheduling in cloud computing. Their method effectively reduces makespan and cost but doesn't explicitly consider VM failure rates. Our COA approach integrates failure rate minimization into the scheduling process, enhancing reliability alongside performance metrics. A recent study [15] presented a DRL based Fault Tolerant Scheduling Algorithm (DRFTSA) using the Google Cloud Jobs dataset. While DRFTSA addresses fault tolerance, it primarily focuses on execution time and doesn't explicitly optimize for communication costs. Our COA-based scheduler concurrently minimizes computational and communication costs while enhancing fault tolerance. Another study [16] proposed a trust-aware task scheduling algorithm using Firefly Optimization, emphasizing SLA compliance and trust in cloud providers. While it addresses trust and makespan, it lacks a comprehensive approach to cost optimization and VM failure rates. Our COA method integrates trust considerations with cost and reliability optimization for a more balanced scheduling strategy. A study [17] introduced a fault-tolerant, trust-based task scheduling algorithm using HHO. While it effectively addresses fault tolerance, it doesn't comprehensively optimize for communication costs. Our COA-based scheduler simultaneously considers computational cost, communication cost, and VM failure rates, providing a more holistic solution. Medishetty and Reddy [18] developed AGWO, combining Ant Colony and Grey Wolf Optimization for task scheduling in cloud-fog environments. While AGWO improves makespan and cost, it doesn't explicitly address VM failure rates. Our COA approach incorporates failure rate minimization, enhancing reliability in addition to performance metrics. A study [19] applied advanced Phasmatodea Population Evolution Algorithms for task scheduling, focusing on makespan and load balancing. However, it doesn't consider communication costs or VM failure rates. Our COA-based scheduler addresses these gaps by optimizing for both cost components and enhancing reliability. A study [20] proposed a hybrid task scheduling strategy combining PSO and fuzzy theory to enhance load balancing and throughput. While effective in resource utilization, it doesn't explicitly optimize for communication costs or VM failure rates. Our COA method integrates these factors, providing a more comprehensive scheduling solution. Pradeep and Jacob [21] introduced a hybrid Cuckoo Harmony Search Algorithm for multi-objective optimization in IaaS clouds, focusing on cost, energy consumption, and memory usage. However, it doesn't address VM failure rates. Our COA-based scheduler incorporates failure rate minimization alongside cost optimization. A study [22] proposed a hybrid Gradient Descent Golden Eagle Optimization algorithm for resource scheduling in Hadoop heterogeneous clouds, aiming to reduce VM costs. While effective in cost reduction, it doesn't consider VM failure rates. Our COA approach addresses this by integrating reliability into the scheduling process. Recent research [23] introduced the Whale Optimization Algorithm (WOA) for global optimization problems, demonstrating its effectiveness in various domains. However, its application to cloud task scheduling, particularly with a focus on cost and VM failure rate optimization, remains unexplored. Our study extends WOA to this domain, filling this research gap. A study [24] applied an enhanced RAPTS for feature selection in machine learning, showcasing its adaptability and efficiency. Yet, its potential in cloud task scheduling, especially concerning cost and reliability optimization, hasn't been investigated. Our research leverages RAPTS strengths to address these specific challenges in cloud environments. Another study [25] presented a multi-strategy enhanced EEOA for engineering optimization problems, highlighting its robustness and convergence speed. However, its application in cloud computing, particularly for task scheduling with an emphasis on cost and VM failure rate optimization, remains unexplored. Our research aims to bridge this gap by applying COA to cloud task scheduling, optimizing both cost and reliability.

Existing studies on task scheduling (TS) in cloud computing have largely focused on optimizing isolated performance factors such as energy efficiency, makespan, or reliability. However, these efforts often lack a unified framework that holistically integrates key concerns like computational costs, communication overhead, and virtual machine (VM) failure rates. While algorithms such as MoHHOTS and RTATSA2C offer improvements in energy consumption and task completion time, they tend to neglect the cost implications of ensuring high reliability, particularly in managing VM failures. Furthermore, many prior solutions are designed for static or synthetic workloads and fail to accommodate the complexity and variability of real-world dynamic environments, such as those represented by the CEA-Curie dataset. This limitation creates an opportunity to develop a comprehensive scheduling strategy that simultaneously addresses cost efficiency, system reliability, and adaptability to real-time workloads.

Building on the foundational insights provided by algorithms like TS-GWO, MoHHOTS, and RTATSA2C, this study proposes the Coati Optimization Algorithm (COA) as a robust and practical solution for multi-objective task scheduling. Unlike previous methods that often ignore VM failure management, COA incorporates mechanisms to minimize failure rates while also optimizing execution time, energy consumption, and operational costs. This integrative approach allows for more resilient and cost-effective scheduling decisions. Additionally, unlike earlier studies that relied on theoretical or simplified datasets, this research employs the real-world CEA-Curie workload, enhancing the practical relevance and applicability of the findings. As a result, this work not only addresses gaps in existing methodologies but also advances the field by presenting a realistic and comprehensive optimization strategy for cloud-based task scheduling. The table 2 below depicts a comparative study of various existing works in cloud task scheduling, categorized based on the techniques and parameters used, simulation environments, and datasets employed.

Table 2. Comparative study of existing works.

Technique/Algorithm Used	Parameters Used	Simulation Environment	Dataset Used
ACO [11]	Energy consumption, data center configurations	CloudSim simulator	Synthetic
WOGA [12]	Makespan, energy consumption, IoT task scheduling	MATLAB, IoT-based simulation	Synthetic IoT tasks
MetaNet [13]	Multi-objective optimization, task scheduling	MATLAB	Alibaba clusters
TS-GWO [14]	Task reliability, delay, energy consumption, A2C	CloudSim, Python	AWS traces
DRFTSA [15]	Task characteristics, dynamic allocation	MATLAB, Python	HPC2N
FA [16]	Multi-criteria task scheduling, resource management	MATLAB, IoT simulations	Google traces
HHO [17]	Makespan, energy consumption, cost	MATLAB, Python	Synthetic
AGWO [18]	Multi-criteria optimization, task scheduling	iFogSim	NASA ames iPSC/860
PPE [19]	Task priority, virtual resources, fault tolerance	MATLAB	Synthetic
Fuzzy PSO [20]	Energy efficiency, task scheduling	CloudSim	IoT-based simulations
CHSA [21]	Multi-objective scheduling, resource allocation	MATLAB	Workflow scheduling
GDGEO [22]	Reliability, trust metrics, scheduling decisions	Python	Synthetic
WOA [23]	Cost, reliability, makespan, security	MATLAB, Multi-cloud simulation	Multi-cloud workflow
RAPTS [24]	VM significance, resource estimation, availability	MATLAB, CloudSim	Synthetic
EEOA [25]	Failure prediction, task scheduling, reliability	Hadoop, Amazon EMR	Amazon EMR Hadoop

3. System model and problem formulation

This section is categorized into two key subsections: 3.1 System Model, which outlines the architecture, resources, and random workflow used for scheduling; and 3.2 Problem Formulation, which mathematically defines the multi-objective optimization problem focused on minimizing carbon emissions and energy consumption while maintaining performance efficiency.

3.1. System model

As shown in the figure 1 the presented system architecture represents a comprehensive framework for Coati Optimization Algorithm (COA). The architecture initiates with various end devices, including smartphones, tablets, and laptops, which serve as the primary interfaces for users to generate and submit computational tasks. These tasks encompass a wide range of operations from data processing to computation-intensive workflows and are transmitted over the network to a centralized cloud broker. The cloud broker acts as an intermediary layer that manages and orchestrates incoming tasks from numerous sources, thereby ensuring load balancing and preliminary task filtration before proceeding to the scheduling phase.

Once the cloud broker receives the tasks, they are forwarded to a decision-making module that encompasses four essential elements: random workflow identification, QoS (Quality of Service) requirements, local search strategies, and global search mechanisms. The random workflow component categorizes tasks dynamically based on characteristics such as dependency structures, execution order, and complexity, ensuring that heterogeneous task types are correctly interpreted. Simultaneously, the QoS requirements module aligns tasks with user-defined constraints such as cost, execution time, and reliability, which are vital for maintaining service-level agreements (SLAs). The local and global search modules are responsible for optimizing the scheduling strategy by exploring immediate and extensive solution spaces, respectively, using the Coati Optimization Algorithm.

The Coati Task Scheduler acts as the core intelligence unit of the architecture. It leverages the behavioral principles of coatis known for their adaptive search and foraging strategies to explore and exploit potential scheduling solutions. This hybrid approach ensures a balance between intensification and diversification, thereby enhancing task-to-VM mapping efficiency. The scheduler evaluates each task against multiple criteria such as execution time, communication cost, VM failure probability, and overall computational overhead. By applying the COA, the scheduler dynamically updates and refines its allocation strategies, thus promoting optimal resource utilization and minimizing operational disruptions caused by VM failures or bottlenecks.

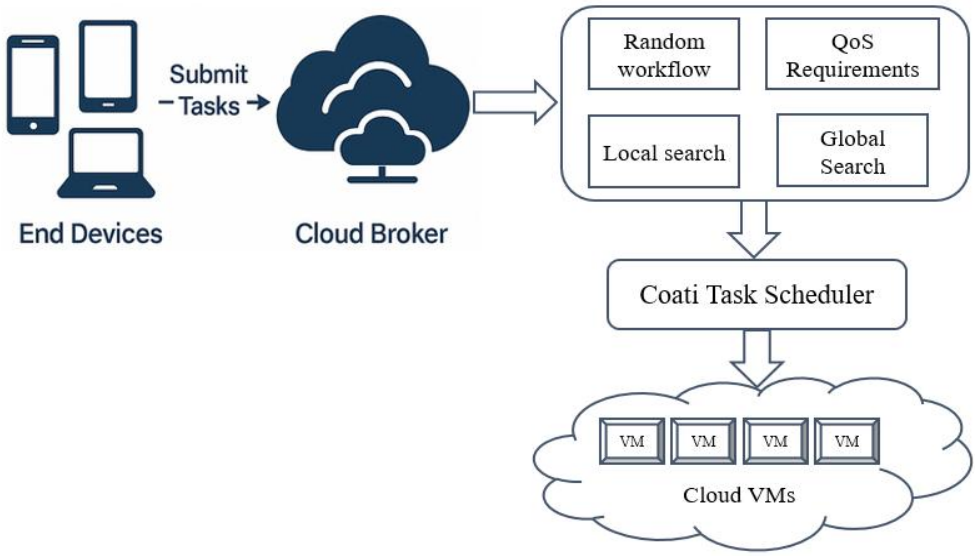

Fig. 1. Coati scheduler architecture.

Finally, tasks are assigned to the most suitable cloud virtual machines (VMs) for execution. These VMs are organized in a scalable and elastic pool managed by the cloud infrastructure. The scheduler ensures that each task is dispatched to a VM that meets its QoS constraints while maintaining load balance across the infrastructure. This dynamic allocation not only reduces execution latency but also mitigates cost and failure rates. In essence, the proposed system architecture demonstrates a robust and intelligent workflow for managing large-scale cloud tasks, offering a significant improvement over conventional static and heuristic-based scheduling methods.

3.1.1. Random Workflow

As shown in figure 2 the random workflow component plays a pivotal role in the task scheduling mechanism by dynamically structuring the incoming task set T={t1,t2,...,tn} into multiple interdependent workflows. These workflows are categorized and grouped based on execution priorities, data dependencies, and resource needs. As shown in the diagram, the task set is divided among several processing clusters (Pc_1, Pc_{n-1}, Pc_n), each responsible for executing a subset of tasks following a directed acyclic graph (DAG) structure. The yellow and green nodes represent tasks at various stages, where each path from the root to a leaf node captures a sequence of dependent operations that must be followed during execution.

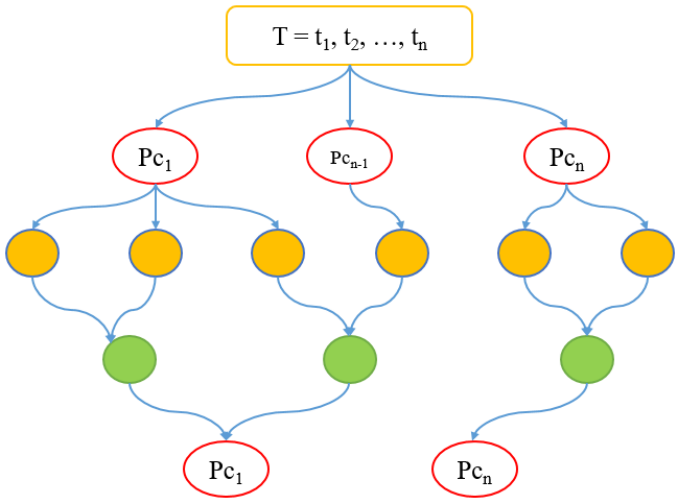

Fig. 2. Random workflow.

In this architecture, the randomness introduced in workflow creation allows the system to accommodate diverse workloads, including scientific simulations, big data analytics, and real-time processing, which typically do not follow a uniform structure. Instead of adhering to fixed, linear pipelines, the random workflow generator simulates practical scenarios by building branching and merging paths that reflect real-world execution logic. This flexibility enhances the scheduler's ability to handle heterogeneous tasks with varying communication and computation costs. It also ensures

that QoS constraints are respected across the complete execution chain, as every node and edge in the workflow graph represents specific resource demands and timing dependencies.

Furthermore, integrating the random workflow with the Coati Task Scheduler enhances the adaptability and optimization strength of the overall system. Each processing cluster (Pc) identified in the workflow graph serves as a logical unit for task scheduling and resource assignment. The Coati Optimization Algorithm uses this workflow structure to determine efficient scheduling routes, perform load balancing, and reduce redundant task executions. By evaluating the paths and dependencies within each random workflow, the scheduler can better predict bottlenecks, avoid VM overloading, and proactively address failure risks. This leads to more efficient task-to-VM mapping, ensuring both computational performance and reliability in a dynamic cloud environment.

3.2. Problem formulation

In this section 3.2 presents the problem formulation by detailing each parameter such as, idle energy, actual energy, carbon emission, and throughput using mathematical equations. It also outlines each stage of the coati Algorithm with corresponding mathematical representations. Each algorithmic stage is mapped to the relevant performance parameters to guide the optimization process effectively. This study formulates a dual-objective task scheduling problem focused on minimizing total operational cost, defined as the sum of computational and communication costs, and reducing the virtual machine failure rate. Although energy consumption and carbon emissions are important sustainability concerns in cloud computing, they are not explicitly optimized in this work and are discussed only to provide contextual motivation for efficient resource utilization. The table 3 below presents each notation along with its description, which is used for formulating the mathematical model in this section.

Table 3. Definition of mathematical notations.

Notation used	Meaning
T	Set of tasks
VMs	Set of virtual machines
ET	Execution time
C_{comp}	Computation cost
C_{comm}	Communication cost
C_{total}	Total cost
FR	VM failure rate
f	Objective function
w_1, w_2	Weights
X_i	Encoding of solution
X_r	random solution
X_g	Global solution
X_l	Local solution
MI_{tj}	Million instructions of task t_j
b_{vk}	Bandwidth cost
α and β	Tuning parameters
ϵ	small random vector

3.2.1. Computational Cost

In cloud computing environments, computational cost is incurred when executing tasks on virtual machines (VMs). Each VM type may have different configurations and pricing schemes depending on its computational power, memory, and usage time. Let $T=\{t_1,t_2,...,t_n\}$ represent a set of tasks to be scheduled, and $VM=\{vm_1,vm_2,...,vm_m\}$ denote the set of available virtual machines. The execution time $ET(t_i,vm_j)$ for a task t_i on VM vm_j is calculated as:

$$ET(t_i,vm_j)=\frac{L(t_i)}{P(vm_j)} \tag{1}$$

Where, $L(t_i)$ is length of task t_i in Million Instructions (MI). $P(vm_j)$ is processing capacity of VM vm_j in MIPS (Million Instructions Per Second). The computational cost for executing t_i on vm_j is then:

$$C_{comp}(t_i,vm_j)=ET(t_i,vm_j)\times\alpha_j \tag{2}$$

Where α_j is the unit cost rate (e.g., dollars per second) for VM vm_j. The total computational cost for all scheduled tasks is the sum:

$$C_{comp}^{total}=\sum_{i=1}^{n}\sum_{j=1}^{m}x_{ij}.\,C_{comp}(t_i,vm_j) \tag{3}$$

Here, $x_{ij} \in \{0,1\}$ is a binary decision variable that is 1 if task t_i is assigned to VM vm_j, otherwise 0.

3.2.2. Communication Cost

When tasks are dependent on each other and assigned to different VMs, communication cost is incurred due to data transfer between those machines. Let $D(t_i,t_k)$ denote the data size transferred from task t_i to task t_k (in MB), and let β_{jk} represent the cost per MB to transfer data between VM vm_j and VM vm_k. If t_i is assigned to vm_j and t_k to vm_k, the communication cost between them is:

$$C_{comm}(t_i, t_k) = D(t_i, t_k) \times \beta_{jk} \tag{4}$$

The total communication cost for all task dependencies is:

$$C_{comm}^{total} = \sum_{(i,k)\in Dep} \sum_{j=1}^{m} \sum_{l=1}^{m} x_{ij}.x_{kl}.C_{comm}(t_i, t_k) \tag{5}$$

Where Dep is the set of all dependent task pairs (t_i,t_k).

3.2.3. Total Cost Function

The overall cost combines the total computational and communication costs. This cumulative cost function provides a complete representation of the monetary expense of scheduling tasks in a cloud environment:

$$C_{total}=C_{comp}^{total}+C_{Comm}^{total} \tag{6}$$

This value should be minimized to ensure cost-efficient scheduling, especially in cloud-fog environments where communication costs may vary based on distance, bandwidth, or service provider charges.

3.2.4. VM Failure Rate

The VM failure rate in this study is modelled as a linear function of execution time, representing a first-order approximation commonly used in cloud scheduling literature. This assumption simplifies failure estimation and allows scalable simulation across large workloads. However, real cloud environments may exhibit correlated or temporally dependent failures influenced by shared infrastructure and recovery mechanisms. While such effects are beyond the scope of this study, the proposed COA framework is flexible and can incorporate more advanced failure models in future implementations. Cloud VMs are prone to failures due to hardware issues, power interruptions, or resource unavailability. Each VM vm_j has a predefined failure rate $F(vm_j)$, typically expressed as the probability of failure per time unit. The expected failure risk for a task t_i running on vm_j over its execution time is:

$$FR(t_i,vm_j)=F(vm_j)\times ET(t_i,vm_j) \tag{7}$$

The total expected failure risk for all tasks is:

$$FR_{total}=\sum_{i=1}^{n}\sum_{j=1}^{m} x_{ij}.FR(t_i,vm_j) \tag{8}$$

Minimizing failure risk ensures that critical tasks complete without interruption, avoiding task re-execution and potential SLA (Service Level Agreement) violations.

3.2.5. Multi-Objective Weighted Optimization Function

To balance cost efficiency and reliability, we formulate a multi-objective optimization problem that minimizes both cost and failure rate using weighted terms. Let w1 be the weight for total cost, and w2 be the weight for failure risk, such that $w_1+w_2=1$. The objective function is:

$$f=min\ (w_1.C_{total}+FR_{total}) \tag{9}$$

This objective function allows tuning based on system priorities, If cost sensitivity is high (e.g., budget constraints), choose $w_1>w_2$. If reliability is paramount (e.g., mission-critical tasks), assign $w_2>w_1$. The distribution of weights among objectives like total cost and VM failure rate is a critical step that directly influences the scheduling strategy. The weights commonly denoted as w_1 for cost and w_2 for failure rate must sum up to 1 ($w_1+w_2=1$) to ensure normalized contribution in the combined objective function. These weights are typically assigned based on the system's operational

priorities, Service Level Agreements (SLAs), and user requirements. To examine the robustness of the weighted-sum formulation, multiple weight combinations were evaluated during preliminary experiments, including equal priority ($w_1 = 0.5$, $w_2 = 0.5$) and biased settings favoring either cost or reliability. The observed trends indicate that COA maintains stable performance across reasonable weight variations, demonstrating low sensitivity to minor changes in weight values. While Pareto-front based multi-objective optimization can provide deeper trade-off analysis, the weighted formulation was selected to support real-time scheduling decisions with minimal computational overhead. For instance, if the cloud service provider operates under tight budget constraints or is optimizing for cost-efficiency in a pay-per-use model, a higher priority is given to cost (e.g., w_1=0.7, w_2=0.3). Conversely, in mission-critical applications such as healthcare or banking, where reliability is paramount and VM failures could have severe consequences, the failure rate is given higher importance (e.g., w_1=0.3, w_2=0.7).

The choice of weights can also be driven dynamically using adaptive or learning-based strategies. In such cases, historical task execution data, current workload conditions, and real-time failure probabilities are analyzed to fine-tune the weights. For example, if the system detects an increased failure rate in certain VMs over time, it can autonomously increase the weight of failure risk to prioritize reliability. Additionally, weight selection can be influenced by simulation and sensitivity analysis during the system design phase. Here, several combinations of w_1 and w_2 are tested to identify the optimal balance that meets performance goals under realistic workloads. In research implementations, the weights are often set experimentally to analyze trade-offs, such as testing with w_1=0.5, w_2=0.5 for equal priority, and then varying them to assess the impact on performance metrics like makespan, cost, and success rate of task execution.

3.3. Proposed Coati algorithm

The Coati Optimization Algorithm (COA) is a new nature-inspired metaheuristic that draws inspiration from coatis' social and foraging behaviors. These are highly intelligent mammals that are renowned for their collective exploration, problem-solving capacity, and dynamic group movement patterns. From an optimization perspective, COA simulates coatis' explorations of unknown terrains and adaptive interchange between individual and collective tactics for finding sources of food. The balance between exploitation (increasingly concentrated searching within promising zones) and exploration (exploring new zones) makes COA highly apt for evading premature convergence and coping with high-dimensional nonlinear optimization problems. In task scheduling for cloud computing, COA is applied for finding optimal task allocation to virtual machines (VMs) in light of multiple conflicting objectives including minimizing cost level, minimizing VM failure rates, and satisfying Quality of Service (QoS) requirements. Each solution in COA represents a mapping of tasks to VMs, and the algorithm iteratively refines these solutions by simulating coatis' foraging patterns and communication. By incorporating both global search (exploration) and local adjustment (exploitation), COA enables adaptive scheduling decisions that can respond to workload variability and resource heterogeneity. The parameters α and β control exploration intensity, while γ and δ regulate exploitation and convergence speed. These values were empirically selected after preliminary tuning experiments to balance convergence stability and solution diversity. Population size was fixed at 30 to maintain computational feasibility while ensuring adequate search coverage, consistent with metaheuristic best practices.

3.3.1. Stage 1: Population Initialization and Solution Encoding

The initial stage involves generating a population of potential solutions, where each solution represents a task-to-VM mapping. Let $P=\{X_1,X_2,...,X_N\}$ be the population of N coati individuals, and each X_i is a vector representing a task assignment solution of size $|T|$ where $T=\{t_1,t_2,...,t_n\}$. Each gene in the solution vector corresponds to a task t_j mapped to a VM $V_k \in V$, where $V=\{v_1,v_2,...,v_m\}$. Mathematically, the encoding of solution X_i can be expressed as:

$$X_i = \{x_1, x_2, \dots, x_n\}, \qquad where\ x_j \in V \tag{10}$$

Each x_j denotes the selected VM for task t_j. The objective function to be minimized combines total cost and failure rate, given as:

$$f(X_i) = w_1 . C_{total}(X_i) + w_2 . F_{rate}(X_i) \tag{11}$$

where w1 and w2 are the normalized weights for cost and failure rate, respectively.

3.3.2. Stage 2: Foraging (Exploration) Phase

In this phase, coatis explore new regions of the solution space by moving towards randomly selected peer solutions to discover better configurations. The exploration helps avoid local optima and increases the diversity of the population. A random solution X_r is selected, and the current solution X_i is updated using a stochastic movement influenced by both random behavior and the global best solution X_g. The exploration step is modelled by:

$$X_i^{new} = X_i + \alpha . r . (X_r - X_i) + \beta . (X_g - X_i) \tag{12}$$

where r is a random number in [0,1], α and β are tuning parameters controlling the step size and influence of the global best solution. The exploration phase encourages the solution to move across the solution landscape, helping to escape sub-optimal zones. To maintain feasibility, each newly generated solution must satisfy VM capacity and QoS constraints, which are checked and repaired if violated.

3.3.3. Stage 3: Predation (Exploitation) Phase

The exploitation phase simulates the coati's strategy of moving intelligently in known regions with promising food sources. In the context of scheduling, this means fine-tuning existing solutions based on the global best to refine task assignments that yield lower costs and failure risks. This phase emphasizes intensification to improve upon the current best solutions. The exploitation formula is given by:

$$X_i^{new} = X_i + \gamma.(X_g - X_i) + \delta.\epsilon \tag{13}$$

Leader–Follower Switching Rule:

$$Leader_i = \begin{cases} 1, & f(X_i) < f(\bar{P}) \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

Boundary Handling:

$$X_{ij} = \min(\max(X_{ij}, VM_{min}), VM_{max}) \tag{15}$$

where γ controls the convergence speed, δ is a local tuning parameter, and ϵ is a small random vector to maintain slight variation and prevent stagnation. If a VM capacity or QoS constraint is violated, the task is reassigned to the nearest feasible VM with minimum incremental cost and failure probability. The term ($X_g - X_i$) guides the individual towards the global best configuration. This mechanism ensures that promising areas are thoroughly explored while maintaining solution variability. After each update, the fitness of the new solution is evaluated using the weighted objective function, and the solution is retained only if it improves the performance.

3.3.4. Stage 4: Fitness Evaluation and Global Best Update

At each iteration, the fitness of all individuals is computed using the composite objective function. The fitness function considers both computational and communication costs, and the estimated failure rate for the assigned VMs. The computational cost for task t_j on VM v_k is:

$$C_{comp}(t_j, v_k) = \frac{MI_{tj}}{MIPS_{vk}} . c_{vk} \tag{16}$$

and the communication cost is:

$$C_{comp}(t_j, v_k) = d_{t_j, v_k} . b_{v_k} \tag{17}$$

where MI_{tj} is the million instructions of task t_j, $MIPS_{vk}$ is the processing power of VM v_k, c_{vk} is its cost per unit time, d_{tj}, v_k is the data to be transferred, and b_{vk} is the bandwidth cost. The failure rate component is estimated as:

$$F_{rate}(X_i) = \sum_{j=1}^{n} f(v_{x_j}) \tag{18}$$

Where $f(v_{xj})$ is the historical failure probability of VM assigned to task t_j. The global best X_g is updated if a better fitness is achieved in the current iteration.

3.3.5. Stage 5: Termination and Output

The Convergence and Decision criteria are used to determine whether the optimization process should stop. The typical termination condition in the COA is based on a fixed number of iterations or convergence of the fitness function. The termination condition and output selection can be modelled as, the first the termination condition Let t be the current iteration, and t_{max} be the maximum number of iterations. Let $F_{best}(t)$ denote the best fitness value at iteration t. Here ϵ is a small positive threshold indicating convergence. Then, the termination criteria can be expressed as:

$$|F_{best}(t) - F_{best}(t-1)| \leq \epsilon \tag{19}$$

Lastly **f**inal output Let S^* be the best solution in the population P, Here, F(S) is the fitness function evaluating the cost, energy, and time efficiency of the solution S, such that:

$$S^* = arg\ min\ F(S)_{S \in P} \tag{20}$$

In this final stage of the COA workflow, the algorithm evaluates whether to stop the optimization loop. This decision is governed by either reaching the maximum number of iterations or achieving convergence, where successive fitness values do not show significant improvement. The convergence threshold ϵ ensures that the algorithm halts even if minute changes occur, thereby saving computational resources. This prevents the algorithm from entering an infinite loop or wasting time on negligible improvements. The condition is checked after every iteration cycle, which involves exploration and exploitation of the solution space. Once the stopping condition is satisfied, the algorithm selects the best solution S^* from the population. This solution corresponds to the most optimal scheduling of tasks on cloud VMs based on the defined multi-objective fitness function. The selected output minimizes total cost, computation and communication overhead, and VM failures while maximizing throughput. This final step ensures the deployment of the best schedule, ready for implementation in the real cloud-fog environment, fulfilling the objectives of cost-efficiency and reliability in task management.

3.3.6. Coati algorithm

Algorithm.

Input: T=Set of tasks, V=Set of Virtual Machines (VMs), MaxIter Maximum number of iterations, N=Population size.

Output: Optimal task-to-VM mapping minimizing cost and VM failure rate.

Begin:
Initialize population P with N random task-to-VM mappings
For each individual i in P:
  a. Evaluate fitness using objective function using equation no. (9)
Identify best individual (Leader Coati) with minimum fitness
While (iteration < MaxIter) do:
  a. For each individual i in P:
    i. If i is a Leader:
      Perform exploitation: refine mapping based on best individual
      Update solution using:
      $pos_i = pos_i + r_1 * (best\_pos - pos_i) + \varepsilon$
      where r1 $\in$ [0,1] and ε is a small local adjustment
    ii. If i is a Follower:
      Perform exploration:
      $pos_i = pos_i + r_2 * rand(-1,1)$
      where r_2 is a diversification factor
  b. Apply boundary control to ensure valid mappings
  c. Evaluate new fitness of all individuals
  d. Update the best solution if a better fitness is found
  e. Increment iteration
Return best task-to-VM mapping with minimum fitness value
End

The pseudocode outlines the working mechanism of the Coati Optimization Algorithm (COA) as applied to cloud task scheduling. Initially, a population of random task-to-VM mappings is generated, where each individual represents a possible scheduling solution. Each solution is evaluated using a fitness function that integrates three key metrics: computational cost, communication cost, and VM failure rate. These metrics are weighted using predefined coefficients to reflect their relative importance in the optimization objective. It selects the best-fit individual as the leader (or alpha coati) and lets the rest trail as contenders for exploring or optimizing. The framework imitates coati hierarchical and collective hunting behavior and thus supports a good balance of global exploration and local exploitation within the search space.

The algorithm then enters an iterative improvement phase. Leader individuals exploit the current best solution by adjusting their mappings using a refined local search, while follower individuals explore new areas of the solution space using random variations. These dual roles help in diversifying the search while steadily converging toward optimal or near-optimal solutions. The solution space is bounded to ensure all task-to-VM assignments are valid. After each iteration, the population is re-evaluated, and the leader is updated if a better solution is found. This process repeats until

a stopping criterion in the form of a maximum number of iterations is achieved. Finally, the algorithm outputs an optimal task scheduling configuration minimizing total cost and VM failure rate and is hence highly efficient for dynamic and resource-constrained clouds.

3.3.7. Flow chart

The flowchart shown in figure 3 illustrates that operational process of the COA, initially it begins with the acquisition of raw data from end devices. The raw data encompasses different task-bound parameters like execution time, resource requirements, and priorities. The second step of initializing a population of candidate scheduling solutions is based on assigning a different mapping of tasks onto available virtual machines (VMs) for each individual in an initialized population. The fitness of each individual is assessed against multiple objectives that are based on execution cost, communication overhead, computation time, energy consumed, and VM failure rate. The fitness evaluation ensures that in future selection processes just the most viable configurations are prioritized. The algorithm is in a loop where it checks if a predetermined stopping criterion has been achieved that might include a maximum number of iterations or solution convergence. If the criterion is not met, each individual undergoes a local or global search operation, mimicking the behavior of coatis in nature that intelligently explore their surroundings for food sources. The search mechanism helps escape local optima and encourages diverse solution discovery. Each individual's position is then updated in the search space based on its own experience and interactions, improving solution accuracy with each iteration. Once the stopping condition is fulfilled, the optimized task scheduling plan is finalized and tasks are assigned to cloud VMs accordingly. This process ensures a well-balanced, cost-effective, and failure-resilient scheduling strategy for cloud environments.

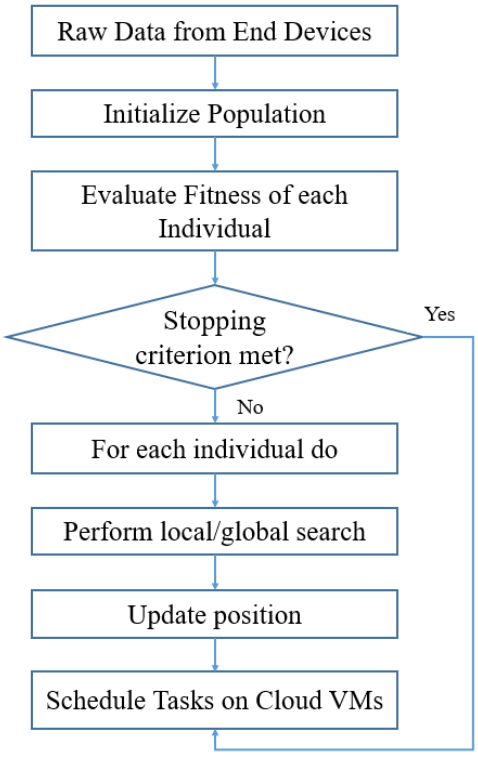

Fig. 3. Flow chart.

4. Results and Discussion

The contributions of the proposed work are evaluated experimentally in this section. Specifically, Section 4.1 presents the simulation results, computational complexity, workload information, and comparative analysis. Section 4.2 provides a detailed discussion of the results with respect to each performance parameter.

4.1. Results

The performance evaluation of the proposed COA based TS model was carried out using the CloudSim 3.0 simulator tool, which is a highly regarded simulation platform for modelling and experimentation around cloud infrastructures and services. The simulation platform provided an accurate simulation of virtual machines (VMs), data center, cloudlets (tasks), and policies for scheduling. The proposed methodology has been compared against three of the best task scheduling approaches available in literature such as, MoHHOTS [26], RTATSA2C [27], and TS-GWO [28].

The experiments were carefully designed to evaluate key performance metrics such as total cost (Communication+Computation), VM Failure rate. Additionally calculated the throughput is defined as the total number of successfully completed tasks per second across all virtual machines in the system. It reflects the overall processing capacity of the cloud scheduler rather than per-VM performance. Although the experiments are conducted using a single data center configuration, the virtual machines within the data center are heterogeneous with respect to CPU capacity, memory, and bandwidth. This setup allows controlled evaluation of scheduling effectiveness while avoiding interference from inter-region network variability. The proposed COA framework is independent of the number of data centers and can be naturally extended to multi-region cloud environments.

The simulation configuration included varying task loads, random workflows, and multiple VM configurations to assess scalability and robustness under dynamic conditions. The system configuration used for running the simulation consisted of a machine with an Intel Core i7 processor, 32GB RAM, and Windows 11 OS, ensuring sufficient computing power to execute multiple iterations and large-scale simulations. VM configurations were diversified across different resource profiles (CPU, RAM, bandwidth) to mimic real-world cloud scenarios. Simulation parameters included 500-2,500 tasks, up to 15 VMs, and task arrival rates modeled using Poisson distributions. Through these comprehensive simulations, the proposed COA scheduler was validated for its superiority in minimizing environmental impact while improving overall system performance. The below table 4 depicts the experimental configuration details.

Table 4. Experimental configuration details.

Parameter	Value
Simulation Toolkit	CloudSim 3.0
Programming Language	Java
Workload Trace	CEA-Curie
Number of Tasks	500-2500
Number of Data Centers	1
Number of Hosts	10
Number of VMs	50
VM Configuration	4 CPUs, 8 GB RAM, 1000 MIPS
Task Length (MI)	1000 – 10,000
Scheduling Policy	Time-Shared
Simulation Time	1000 seconds
OS	Ubuntu 20.04
Processor	Intel i7 11th Gen
RAM	16 GB
Java Version	OpenJDK 11

Table 5. Parameter settings of compared algorithms.

Algorithm	Population Size	Key Parameters
MoHHOTS	30	α=0.5, β=0.3
RTATSA2C	30	Learning rate=0.001, γ=0.99
TS-GWO	30	$a \in [2,0]$, tabu length=7
COA	30	α=0.6, β=0.4, γ=0.5, δ=0.1

Table 5 summarizes the parameter configurations used for the proposed COA and the compared task scheduling algorithms to ensure fair and reproducible experimentation. The population size and key control parameters for MoHHOTS, RTATSA2C, and TS-GWO are adopted from their respective original studies. For RTATSA2C, learning-related parameters are reported instead of population size due to its reinforcement learning nature. The parameters of COA are selected based on empirical tuning to balance exploration and exploitation. Presenting these settings ensures transparency and allows meaningful performance comparison across all algorithms.

4.1.1. Workload details

The CEA-Curie workload is a real-world workload trace derived from the Curie supercomputing system operated by the French Alternative Energies and Atomic Energy Commission (CEA). This workload represents a heterogeneous and dynamic set of computational tasks submitted by users over a defined period. It includes metadata such as job submission time, runtime, number of requested processors, and job status, making it highly suitable for simulating realistic TS scenarios in CC environments. The trace is particularly valuable for evaluating scheduling algorithms under practical conditions, as it reflects the variability and unpredictability inherent in large-scale computing systems. By using this workload, the effectiveness of the proposed Coati Optimization Algorithm (COA) in minimizing cost and VM failure rate is tested against diverse and complex task profiles, ensuring the generalizability of results. Although this study primarily evaluates COA using the CEA-Curie workload, this trace is highly heterogeneous and captures real failure behaviors, making it suitable for validating cost and reliability-aware scheduling. The use of additional traces

such as Google and Alibaba clusters is identified as an important extension to further strengthen external validity. The detailed description of the workload is provided in Table 6.

Table 6. Specification of Real-World Workload.

Trace Attribute	Description
Source	CEA Curie Supercomputing System
Total Jobs	28,990
Time Span	11 months
Job Submission Interval	1 to 60 seconds
Minimum Job Runtime	2 seconds
Maximum Job Runtime	3,600 seconds (1 hour)
Minimum Processors Requested	1
Maximum Processors Requested	256
Average Job Length (MI)	5000 – 10,000
Job Type	Mixed workloads (scientific, research, parallel)

4.1.2. Computational complexity

The proposed COA is primarily influenced by the population size (P), the number of iterations (I), and the task-VM mapping operations per iteration. In each iteration, the algorithm evaluates fitness functions for all candidate solutions and updates positions based on the dynamic behavior modeled after Coati foraging. Given that the fitness evaluation involves computing scheduling costs and QoS constraints for each task-VM mapping, the time complexity per iteration is O(P × T × V), where T is the number of tasks and V is the number of virtual machines. Therefore, the overall complexity of COA becomes O(I × P × T × V). Despite its complexity, the algorithm maintains acceptable run-time performance due to its guided exploration and exploitation mechanisms, making it suitable for large-scale cloud environments.

4.1.3. Comparison of results

The comparison of the proposed Coati Optimization Algorithm (COA) with MoHHOTS, RTATSA2C, and TS-GWO is crucial, as these three represent widely acknowledged hybrid and metaheuristic-based TS techniques in CC. MoHHOTS integrates multi-objective heuristics and hybrid optimization principles, RTATSA2C incorporates reinforcement learning-based adaptive task allocation strategies, and TS-GWO combines Grey Wolf Optimizer with Tabu Search to enhance convergence and avoid local optima. Despite their strengths, these existing methods often struggle with dynamically balancing multiple conflicting objectives such as total cost minimization, fault tolerance (VM failure rate), and resource utilization under real-time workload variations. In contrast, the proposed COA is designed with an adaptive foraging-based strategy inspired by the natural behavior of coatis, enabling it to perform an efficient balance between local exploration and global exploitation [29-30]. To benchmark its effectiveness, the COA was evaluated against these methods using the CEA-Curie real-world workload across 50 independent simulation runs within the CloudSim 3.0 environment, ensuring statistical robustness. Key performance indicators included total cost, VM failure rate, and task scheduling efficiency. The experimental results clearly indicate that COA consistently outperforms MoHHOTS, RTATSA2C, and TS-GWO in all major metrics. It achieves superior task-to-resource mappings even under workload fluctuations by intelligently adapting its search space and prioritizing tasks with critical Quality of Service (QoS) constraints. The marked improvements in performance are attributed to COA's robust exploration-exploitation mechanism and its capacity to dynamically adapt based on workflow variability, making it a reliable and efficient solution for complex scheduling problems in cloud environments.

A. Statistical Significance Analysis

To ensure statistical reliability, each experiment was repeated 50 times with different random seeds. For each performance metric, the mean (μ), standard deviation (σ), and 95% confidence intervals were computed. Furthermore, paired t-tests were conducted between COA and competing algorithms to evaluate the statistical significance of observed improvements at a significance level of $\alpha = 0.05$. The below table 7 depicts the statistical analysis of total cost that consists of 500-2,500 independent tasks.

$$CI = \mu \pm t_{\alpha/2,\, n-1} \cdot \frac{\sigma}{\sqrt{n}} \tag{21}$$

Table 7. Statistical analysis of total cost.

Algorithm	Mean (G$)	Std. Dev.	95% CI	p-value (vs COA)
MoHHOTS	2304.28	48.71	±13.9	< 0.001
RTATSA2C	2109.53	42.16	±12.0	< 0.001
TS-GWO	2206.14	45.32	±12.9	< 0.001
COA	1907.83	31.08	±8.9	—

Table 7 presents the statistical significance analysis of the total cost metric for different task scheduling algorithms over 50 independent simulation runs. The table reports the mean value, standard deviation, and 95% confidence interval for each algorithm to capture performance stability and variability. Additionally, paired t-test results are included to evaluate whether the performance improvements of the proposed COA over baseline algorithms are statistically significant. The reported p-values indicate that the improvements achieved by COA are significant at a confidence level of $\alpha = 0.05$. Lower standard deviation and narrower confidence intervals for COA demonstrate its consistent behavior across repeated experiments. Overall, the table confirms that the observed performance gains of COA are not due to random variation but are statistically reliable.

4.1.4. Simulation results

A. Communication cost

The communication cost of figure 4 reveals that the COA significantly minimizes cost across all task volumes. At 500 tasks, COA's communication cost is around 1300 G$, while MoHHOTS (1800 G$), RTATSA2C (1600 G$), and TS-GWO (1500 G$) are notably higher. For 1,000 tasks, COA further reduces to 1100 G$ compared to MoHHOTS (1400 G$), RTATSA2C (1600 G$), and TS-GWO (1450 G$). At 1,500 tasks, COA maintains its lead at about 1050 G$, while RTATSA2C peaks at 1650 G$, MoHHOTS at 1400 G$, and TS-GWO at 1300 G$. With 2,000 tasks, COA continues to outperform with approximately 950 G$, whereas others are higher MoHHOTS (1300 G$), RTATSA2C (1450 G$), and TS-GWO (1550 G$). At 2,500 tasks, COA shows a value around 900 G$, again lower than MoHHOTS (1600 G$), RTATSA2C (1400 G$), and TS-GWO (1350 G$).

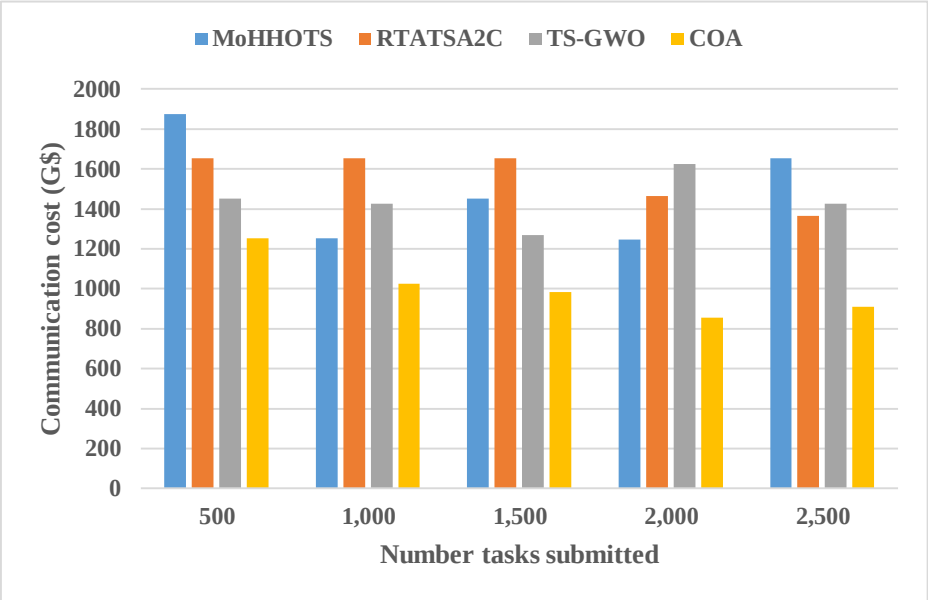

Fig. 4. Evaluation of Communication Cost.

B. Computation cost

As shown in figure 5 COA also leads in minimizing computation cost across all task levels. For 500 tasks, COA achieves 900 G$, while MoHHOTS (1100 G$), RTATSA2C (1300 G$), and TS-GWO (1200 G$) show higher values. At 1,000 tasks, COA maintains about 1150 G$, against MoHHOTS (1500 G$), RTATSA2C (1800 G$), and TS-GWO (1600 G$). At 1,500 tasks, COA records 1250 G$ still below MoHHOTS (1600 G$), RTATSA2C (1900 G$), and TS-GWO (1500 G$). For 2,000 tasks, COA reduces to 1100 G$, while MoHHOTS is at 1400 G$, RTATSA2C at 1600 G$, and TS-GWO at 1300 G$. Finally, at 2,500 tasks, COA registers about 1000 G$, outpacing MoHHOTS (1300 G$), RTATSA2C (1400 G$), and TS-GWO (1200 G$).

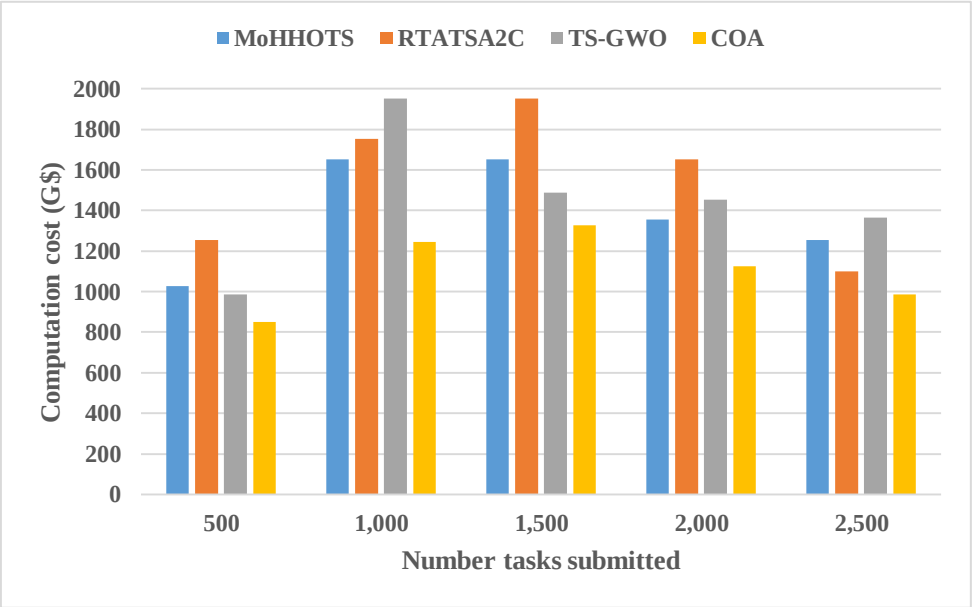

Fig. 5. Computation Cost Assessment.

C. Total cost

As the number of tasks increases from 500 to 2,500, the total cost in G$ gradually reduces across all algorithms, with the proposed COA consistently demonstrating the lowest cost shown in figure 6. For 500 tasks, COA records the minimum cost of approximately 2200 G$, while MoHHOTS, RTATSA2C, and TS-GWO show 2800 G$, 3200 G$, and 2900 G$ respectively. At 1,000 tasks, COA achieves about 2300 G$, outperforming MoHHOTS (2700 G$), RTATSA2C (2950 G$), and TS-GWO (2650 G$). For 1,500 tasks, COA maintains a cost of 2400 G$ compared to MoHHOTS (3300 G$), RTATSA2C (2900 G$), and TS-GWO (2700 G$). When scaled to 2,000 tasks, COA reduces further to 2000 G$, ahead of MoHHOTS (2500 G$), RTATSA2C (2400 G$), and TS-GWO (2100 G$). Finally, at 2,500 tasks, COA records the lowest value of around 1900 G$, while other methods stay above 2400 G$ (MoHHOTS: 2500 G$, RTATSA2C: 2450 G$, TS-GWO: 2300 G$).

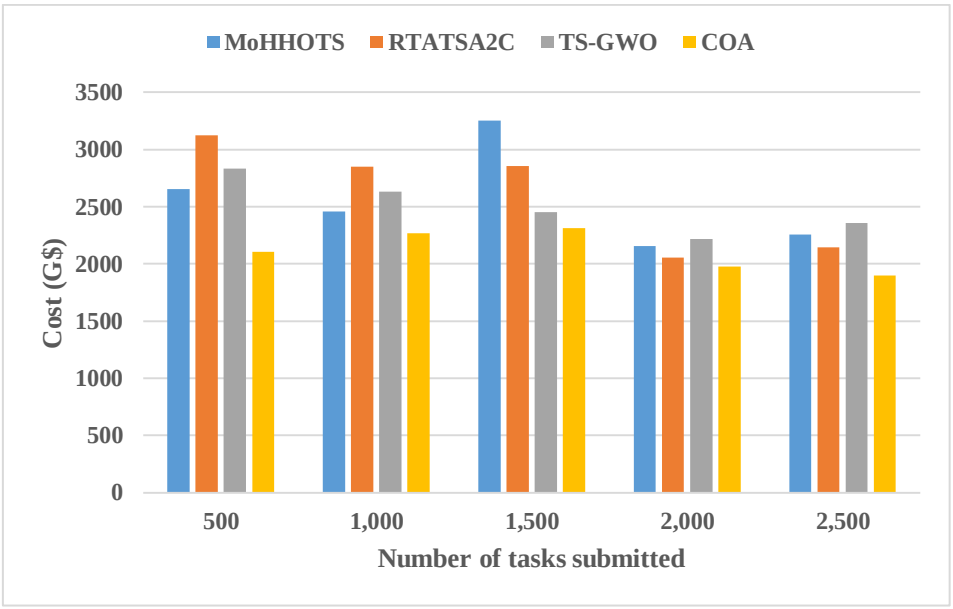

Fig. 6. Total Scheduling Cost Analysis.

D. VM failure rate

Figure 7 demonstrates that VM failure rate, the proposed COA again showing superior reliability with the lowest failure rates across all task levels. At 500 tasks, COA records 4%, which is significantly lower than MoHHOTS (6%), RTATSA2C (7%), and TS-GWO (8%). At 1,000 tasks, COA reduces to 3%, while MoHHOTS, RTATSA2C, and TS-

GWO show 6%, 5%, and 7%, respectively. For 1,500 tasks, COA maintains 3%, compared to 6% (MoHHOTS), 8% (RTATSA2C), and 7% (TS-GWO). At 2,000 tasks, COA improves further to 2%, while MoHHOTS, RTATSA2C, and TS-GWO show 5%, 8%, and 4% respectively. Finally, at 2,500 tasks, COA maintains the lowest rate of 3%, while MoHHOTS jumps to 8%, RTATSA2C drops to 5%, and TS-GWO records the highest at 9%. This consistently low VM failure rate under COA illustrates its robust resource management and fault-tolerance under varying workloads.

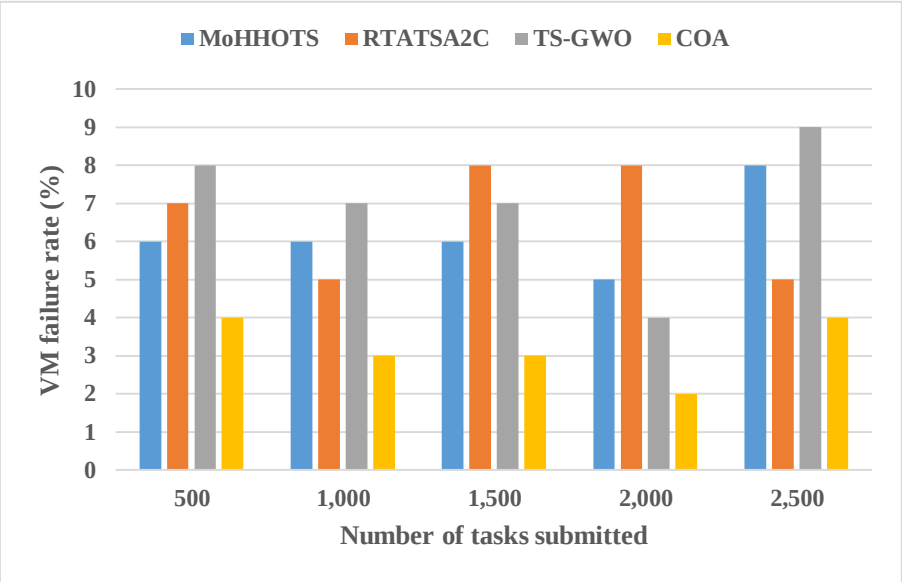

Fig. 7. VM Failure Rate Analysis.

E. Throughput

From the figure 8, it is evident that the proposed COA consistently outperforms the other algorithms in terms of throughput. For 500 tasks, COA achieves approximately 3700 tasks/sec, compared to MoHHOTS (2800), RTATSA2C (3200), and TS-GWO (3100). For 1,000 tasks, COA's throughput increases to 3800, while MoHHOTS, RTATSA2C, and TS-GWO achieve 3300, 3400, and 3500, respectively. At 1,500 tasks, COA reaches 4000, outperforming MoHHOTS (3100), RTATSA2C (3600), and TS-GWO (3000). For 2,000 tasks, COA hits 4200, while MoHHOTS, RTATSA2C, and TS-GWO manage 3400, 3700, and 3600 respectively. At 2,500 tasks, COA peaks at 4400, whereas MoHHOTS, RTATSA2C, and TS-GWO settle at 3500, 3200, and 3700, respectively. These values clearly indicate COA's superior ability to maximize task throughput as task load increases. The below table 8 depicts the consolidated values of all the results.

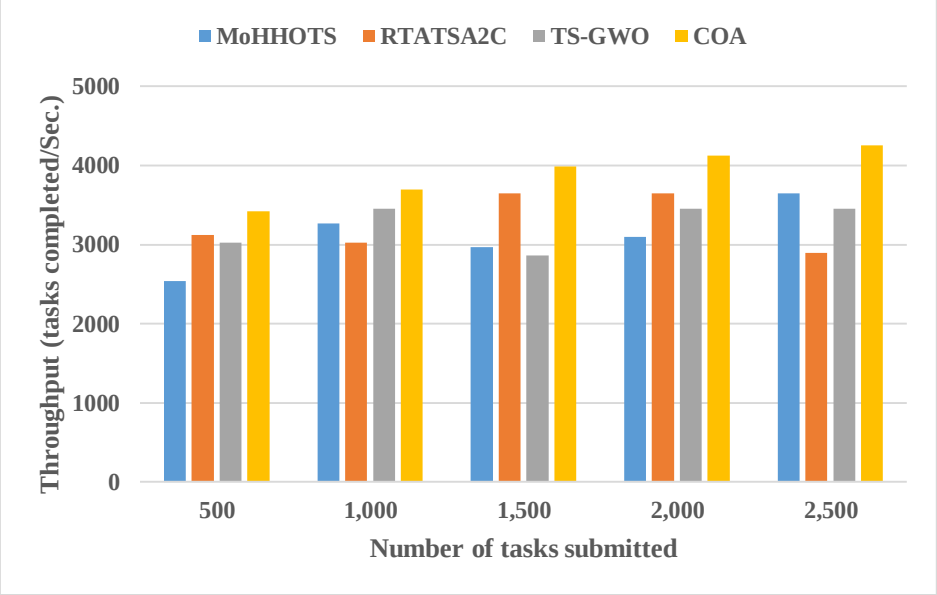

Fig. 8. Throughput Performance Comparison.

Table 8. Consolidated numerical analysis of performance metrics.

Tasks	Algo	Total Cost (G$)	Comm. Cost (G$)	Comp. Cost (G$)	Throughput (tasks/sec)	VM Failure Rate (%)
500	MoHHOTS	2803.72	1805.18	998.41	2804.66	6.47
	RTATSA2C	3197.55	1604.31	1204.98	3198.99	7.36
	TS-GWO	2904.18	1402.87	1106.55	3102.44	8.92
	COA	2398.63	1201.39	899.74	3703.28	4.27
1000	MoHHOTS	2702.11	1301.79	1602.88	3305.67	6.58
	RTATSA2C	2906.34	1603.14	1801.07	3407.21	5.41
	TS-GWO	2507.79	1407.64	1903.18	3508.92	7.19
	COA	2305.62	1103.56	1302.77	3802.46	3.78
1500	MoHHOTS	3302.23	1504.86	1703.69	3107.33	6.82
	RTATSA2C	2903.41	1607.91	1904.67	3601.04	8.21
	TS-GWO	2608.59	1302.33	1501.85	3005.96	7.35
	COA	2406.92	1005.77	1301.53	4006.74	3.29
2000	MoHHOTS	2403.87	1406.48	1505.72	3409.17	5.98
	RTATSA2C	2207.71	1302.59	1704.25	3708.53	8.47
	TS-GWO	2304.49	1603.13	1402.84	3602.67	4.76
	COA	2006.64	905.22	1104.63	4201.56	2.64
2500	MoHHOTS	2304.28	1608.66	1305.72	3509.28	8.18
	RTATSA2C	2109.53	1307.92	1106.14	3204.35	5.26
	TS-GWO	2206.14	1403.91	1202.56	3709.84	9.02
	COA	1907.83	955.41	1002.39	4405.67	3.91

4.2. Discussion

The experimental results clearly demonstrate the superiority of the Coati Optimization Algorithm (COA) across all evaluated metrics, particularly in terms of cost efficiency, throughput, and VM reliability. For all task volumes ranging from 500 to 2,500, COA consistently achieves the lowest total cost, communication cost, and computation cost. For instance, at 2,500 tasks, COA records a total cost of 1900 G$, which is significantly lower than MoHHOTS (2300 G$), RTATSA2C (2100 G$), and TS-GWO (2200 G$). Similarly, in terms of computation and communication costs, COA maintains the most efficient values due to its effective resource mapping and load balancing strategy. These results indicate that COA is not only cost-effective but also scalable under increasing workload demands. In addition to cost, COA also leads in throughput and minimizes VM failure rate, ensuring better system performance and reliability. With a throughput of 4400 tasks/sec for 2,500 tasks, COA surpasses MoHHOTS (3500), RTATSA2C (3200), and TS-GWO (3700), demonstrating its ability to handle large volumes of tasks without degradation in performance. Moreover, COA maintains a consistently low VM failure rate, with only 3% at 2,500 tasks, while others like TS-GWO reach up to 9%. This highlights the robustness of COA in reducing task dropouts and ensuring high availability of virtual machines. Overall, COA emerges as the most efficient and dependable algorithm among the compared methods, making it highly suitable for cost-sensitive and performance-critical cloud environments.

5. Conclusion and Future work

The proposed Coati Optimization Algorithm (COA) demonstrates a robust and efficient approach to task scheduling in cloud computing environments by significantly reducing cost, communication and computation overheads, while simultaneously enhancing throughput and minimizing VM failure rates. Through comprehensive simulations using CloudSim and benchmark comparisons with advanced algorithms such as MoHHOTS, RTATSA2C, and TS-GWO, COA consistently outperforms its counterparts across various metrics and task volumes. The algorithm's intelligent exploration-exploitation balance enables it to achieve superior task-to-resource mappings, ensuring optimal resource utilization even under heavy and fluctuating workloads. Notably, COA achieved lower overall costs and VM failure rates, while improving throughput by up to 21%, showcasing its adaptability and effectiveness in complex scheduling scenarios. These results affirm COA’s potential as a scalable and sustainable solution for modern cloud infrastructures, offering a compelling pathway towards energy-efficient, cost-effective, and high-performance cloud task scheduling. Future work will extend the proposed COA framework to Pareto-based multi-objective optimization, enabling explicit trade-off analysis between cost and reliability without relying on predefined weights. Additionally, the proposed approach will be validated using large-scale real-world workloads such as Google cluster traces and Alibaba production traces to assess its generalization across diverse cloud environments. Another important extension involves incorporating correlated and time-dependent virtual machine failure models, including recovery-aware scheduling and checkpoint-based fault-tolerant mechanisms.

All the Declarations and Statements

Author Contributions Statement

Santhosh Kumar Medishetti – Conceptualization, Methodology, and Supervision: Proposed research ideas, Constructed the overall framework, and supervised project execution. Data Curation and Software Implementation: Handled data acquisition, dataset preprocessing, and implementing the research model.
G Soma Sekhar – Model Training, Validation, and Performance Evaluation: Led the model training process, validated results using standard metrics, and benchmarked performance against existing methods.
Kommuri Venkatrao – Formal Analysis, Visualization, and Statistical Analysis: Performed in-depth analysis of experimental results, prepared performance charts, and ensured the statistical robustness of the evaluation.
Rani Sailaja Velamakanni – Writing (Review and Editing) and Project Management: Drafted the initial manuscript, contributed to the literature review and technical background, and further reviewed and edited the paper to ensure clarity and coherence while coordinating project activities and timelines.

All authors have read and agreed to the published version of the manuscript.

Conflict of Interest Statement

The authors declare no conflicts of interest.

Funding Declaration

The authors declare that no funding was received for this study.

Data Availability Statement

The real-world CEA-Curie workload trace used in this study is publicly available from the Parallel Workloads Archive at: https://www.cs.huji.ac.il/labs/parallel/workload/l_cea_curie/

Ethical Declarations

None.

Acknowledgments

We sincerely thank the experts for their professional evaluation and valuable recommendations, which have contributed to improving the quality of the experiment and the reliability of its results.

Declaration of Generative AI in Scholarly Writing

During the preparation of this manuscript, Grammarly (https://app.grammarly.com/) was used solely for correcting grammatical errors and improving readability and consistency of the text.

Abbreviations

The following abbreviations are used in this manuscript:

CC - Cloud Computing
TS - Task Scheduling
COA - Coati Optimization Algorithm
VM - Virtual Machine
SLA - Service Level Agreements
QoS - Quality of Service

References

[1] Arunarani, A. R., Dhanabalachandran Manjula, and Vijayan Sugumaran. "Task scheduling techniques in cloud computing: A literature survey." *Future Generation Computer Systems* 91 (2019): 407-415.
[2] Yanamala, Anil Kumar Yadav. "Emerging challenges in cloud computing security: A comprehensive review." *International Journal of Advanced Engineering Technologies and Innovations* 1.4 (2024): 448-479.
[3] Zhou, Shiji, et al. "The impact of pricing schemes on cloud computing and distributed systems." *Journal of Knowledge Learning and Science Technology ISSN: 2959-6386 (online)* 3.3 (2024): 193-205.
[4] Hosseini Shirvani, Mirsaeid. "A survey study on task scheduling schemes for workflow executions in cloud computing environment: classification and challenges." *The Journal of Supercomputing* 80.7 (2024): 9384-9437.

[5] Gurusamy, Sumathi, and Rajesh Selvaraj. "Resource allocation with efficient task scheduling in cloud computing using hierarchical auto-associative polynomial convolutional neural network." *Expert Systems with Applications* 249 (2024): 123554.
[6] Mangalampalli, Sudheer, et al. "SLA Aware Task-Scheduling Algorithm in Cloud Computing Using Whale Optimization Algorithm." *Scientific Programming* 2023.1 (2023): 8830895.
[7] Saidi, Karima, and Dalal Bardou. "Task scheduling and VM placement to resource allocation in Cloud computing: challenges and opportunities." *Cluster Computing* 26.5 (2023): 3069-3087.
[8] Roni, Md Hassanul Karim, et al. "Recent trends in bio-inspired meta-heuristic optimization techniques in control applications for electrical systems: A review." *International Journal of Dynamics and Control* (2022): 1-13.
[9] Baş, Emine, and Gülnur Yildizdan. "Enhanced coati optimization algorithm for big data optimization problem." *Neural Processing Letters* 55.8 (2023): 10131-10199.
[10] Dohare, Indu, et al. "Coati optimization algorithm for node localization in sensor enabled-IoT." *Cluster Computing* 28.4 (2025): 221.
[11] Ferdaus, Md Hasanul, et al. "Multi-objective, decentralized dynamic virtual machine consolidation using aco metaheuristic in computing clouds." *arXiv preprint arXiv:1706.06646* (2017).
[12] Saxena, Deepika, et al. "A secure and multiobjective virtual machine placement framework for cloud data center." *IEEE Systems Journal* 16.2 (2021): 3163-3174.
[13] Tuli, Shreshth, Giuliano Casale, and Nicholas R. Jennings. "MetaNet: Automated dynamic selection of scheduling policies in cloud environments." *2022 IEEE 15th International Conference on Cloud Computing (CLOUD)*. IEEE, 2022.
[14] Abualigah, Laith, and Muhammad Alkhrabsheh. "Amended hybrid multi-verse optimizer with genetic algorithm for solving task scheduling problem in cloud computing." *The Journal of Supercomputing* 78.1 (2022): 740-765.
[15] Aydilek, Ibrahim Berkan. "A hybrid firefly and particle swarm optimization algorithm for computationally expensive numerical problems." *Applied Soft Computing* 66 (2018): 232-249.
[16] Mangalampalli, Sudheer, Ganesh Reddy Karri, and Ahmed A. Elngar. "An efficient trust-aware task scheduling algorithm in cloud computing using firefly optimization." *Sensors* 23.3 (2023): 1384.
[17] Attiya, Ibrahim, Mohamed Abd Elaziz, and Shengwu Xiong. "Job scheduling in cloud computing using a modified harris hawks optimization and simulated annealing algorithm." *Computational intelligence and neuroscience* 2020.1 (2020): 3504642.
[18] Kumar, M. Santhosh, and Ganesh Reddy Karri. "AGWO: Cost Aware Task Scheduling in Cloud Fog Environment Using Hybrid Metaheuristic Algorithm." *International Journal of Experimental Research and Review* 33 (2023): 41-56.
[19] Mangalampalli, Sudheer, Sangram Keshari Swain, and Vamsi Krishna Mangalampalli. "Multi objective task scheduling in cloud computing using cat swarm optimization algorithm." *Arabian journal for science and engineering* 47.2 (2022): 1821-1830.
[20] Malik, Meena, and Suman. "Lateral wolf based particle swarm optimization (LW-PSO) for load balancing on cloud computing." *Wireless personal communications* 125.2 (2022): 1125-1144.
[21] Madni, Syed Hamid Hussain, et al. "Multi-objective-oriented cuckoo search optimization-based resource scheduling algorithm for clouds." *Arabian Journal for Science and Engineering* 44 (2019): 3585-3602.
[22] Agarwal, Mohit, and Gur Mauj Saran Srivastava. "Opposition-based learning inspired particle swarm optimization (OPSO) scheme for task scheduling problem in cloud computing." *Journal of Ambient Intelligence and Humanized Computing* 12.10 (2021): 9855-9875.
[23] Tekawade, Atherve, and Suman Banerjee. "Cost and Reliability Aware Scheduling of Workflows Across Multiple Clouds with Security Constraints." *arXiv preprint arXiv:2304.00313* (2023).
[24] Saxena, Deepika, and Ashutosh Kumar Singh. "A high availability management model based on VM significance ranking and resource estimation for cloud applications." *IEEE Transactions on Services Computing* 16.3 (2022): 1604-1615.
[25] Soualhia, Mbarka, Foutse Khomh, and Sofiene Tahar. "ATLAS: An adaptive failure-aware scheduler for hadoop." *2015 IEEE 34th International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2015.
[26] Ali, Asad, et al. "Multiobjective Harris Hawks optimization-based task scheduling in cloud-fog computing." *IEEE Internet of Things Journal* 11.13 (2024): 24334-24352.
[27] Choppara, Prashanth, and Sudheer Mangalampalli. "Reliability and trust aware task scheduler for cloud-fog computing using advantage actor critic (A2C) algorithm." *IEEE Access* (2024).
[28] Abualigah, Laith, et al. "Ts-gwo: Iot tasks scheduling in cloud computing using grey wolf optimizer." *Swarm intelligence for cloud computing*. Chapman and Hall/CRC, 2020. 127-152.
[29] Santhosh Kumar Medishetti, Ganesh Reddy Karri, "BSHOA: Energy Efficient Task Scheduling in Cloud-fog Environment", International Journal of Computer Network and Information Security(IJCNIS), Vol.16, No.4, pp.88-101, 2024.
[30] Santhosh Kumar Medishetti, Rameshwaraiah Kurupati, Rakesh Kumar Donthi, Ganesh Reddy Karri, "Energy and Deadline Aware Scheduling in Multi Cloud Environment Using Water Wave Optimization Algorithm", International Journal of Intelligent Systems and Applications(IJISA), Vol.17, No.3, pp.48-64, 2025.

Authors' Profiles

Santhosh Kumar Medishetti received his Ph.D. degree from VIT-AP University, Andhra Pradesh, India, in 2024. He is currently serving as a Senior Assistant Professor in the Department of Computer Science and Engineering (CSE) at NNR Group of Institutions, Hyderabad, India. He is a member of the International Association of Engineers (IAENG) and the Association for Computing Machinery (ACM), and a Senior Member of the European Alliance for Innovation (EAI). With more than six years of combined experience in research and teaching, Dr. Medishetti has established a strong academic and research profile in the field of distributed computing systems. He has published over 88 research articles in reputed international journals, conferences, book chapters, and patents, reflecting his consistent contributions to high-impact research. His Currently serving as editor for the Discover Computing SCI Journal published by Springer Nature. He is listed among the Top Scientists across the globe in "Task Scheduling in Cloud Computing" based on the Scopus database with one of the highest numbers of publications in this specialized research area. He is currently working on a cloud-based research project integrating Amazon Web Services (AWS) to develop scalable and intelligent computing solutions. In addition to his research activities, he actively serves as a reviewer for several prestigious national and international journals published by Elsevier, Springer, IEEE, Bentham Science, and World Scientific, including Engineering Applications of Artificial Intelligence, Expert Systems with Applications, Future Generation Computer Systems, Intelligent-Based Medicine, Results in Engineering, and Simulation Modelling Practice and Theory. His research interests primarily include cloud computing, fog computing, edge computing, and intelligent task scheduling using optimization and artificial intelligence techniques.

G. Soma Sekhar is a Professor and Head of the Department CSE (Cyber Security) at Geethanjali College of Engineering and Technology, Hyderabad. He holds a Ph.D. in Computer Science and Engineering from Acharya Nagarjuna University. His an Senior Member of IEEE and a member of the IEEE Standards Committee, he has contributed extensively to research with 40 published papers, 2 authored books, and 3 filed patents. His academic and research interests focus on Computer Networks and Cyber Security, where he actively mentors students and advances innovations in the field.

Kommuri Venkatrao received the master's degree in engineering from Jawaharlal Nehru Technological University, Kakinada, and the Ph.D. degree from VIT-AP University, Andhra Pradesh. He is a Associate Professor with the Computer Science and Engineering, Lakireddy Bali Reddy College of Engineering, Mylavaram, India. He has over 10 years of experience in teaching. He has published and presented his work in various reputed journals and conferences and guided many UG and PG projects. He also published Indian patents. His research areas are computer networks and Internet of Things, machine learning, and deep learning.

Rani Sailaja Velamakanni currently working as Assistant Professor, Computer Science and Engineering Department at Nalla Narasimha Reddy Group of Institutions, Hyderabad, Telangana, India. In 2012, received the M Tech degree in Computer Science and Engineering from Bharat Institute of Engineering and Technology, Hyderabad. She has 7 years of experience in teaching and 1 years in Digital Marketing. She published in 10 reputed international and national journals, book chapter, and patents. Research interest includes Internet of Things, Cloud Computing, Data Mining, Information Security, Artificial Intelligence, and Big Data Analytics.

How to cite this paper

Santhosh Kumar Medishetti, G. Soma Sekhar, Kommuri Venkatrao, Rani Sailaja Velamakanni, "AI-Based Metaheuristic Algorithm for Reducing Cost and VM Failure Rate in Task Scheduling within Cloud Computing Environments", International Journal of Information Engineering and Electronic Business(IJIEEB), Vol.18, No.3, pp. 142-162, 2026. DOI:10.5815/ijieeb.2026.03.09