scientific reports

OPEN

A multi-objective approach to load balancing in cloud environments integrating ACO and WWO techniques

Umesh Kumar Lilhore[1], Sarita Simaiya[1,4]✉, Yogendra Narayan Prajapati[2], Anjani Kumar Rai[3], Ehab Seif Ghith[5], Mehdi Tlija[6], Tarik Lamoudan[7] & Abdelaziz A. Abdelhamid[8,9]✉

Effective load balancing and resource allocation are essential in dynamic cloud computing environments, where the demand for rapidity and continuous service is perpetually increasing. This paper introduces an innovative hybrid optimisation method that combines water wave optimization (WWO) and ant colony optimization (ACO) to tackle these challenges effectively. ACO is acknowledged for its proficiency in conducting local searches effectively, facilitating the swift discovery of high-quality solutions. In contrast, WWO specialises in global exploration, guaranteeing extensive coverage of the solution space. Collectively, these methods harness their distinct advantages to enhance various objectives: decreasing response times, maximising resource efficiency, and lowering operational expenses. We assessed the efficacy of our hybrid methodology by conducting extensive simulations using a cloud-sim simulator and a variety of workload trace files. We assessed our methods in comparison to well-established algorithms, such as WWO, genetic algorithm (GA), spider monkey optimization (SMO), and ACO. Key performance indicators, such as task scheduling duration, execution costs, energy consumption, and resource utilisation, were meticulously assessed. The findings demonstrate that the hybrid WWO-ACO approach enhances task scheduling efficiency by 11%, decreases operational expenses by 8%, and lowers energy usage by 12% relative to conventional methods. In addition, the algorithm consistently achieved an impressive equilibrium in resource allocation, with balance values ranging from 0.87 to 0.95. The results emphasise the hybrid WWO-ACO algorithm's substantial impact on improving system performance and customer satisfaction, thereby demonstrating a significant improvement in cloud computing optimisation techniques.

Keywords Water wave optimization, Hybrid optimization, Ant colony optimization, Cloud load balancing, Resource allocation

Abbreviations

WWO	Water wave optimization
ACO	Ant colony optimization
GA	Genetic algorithm
SMO	Spider monkey optimization
RR	Round robin
FCFS	First-come-first-serve
LC	Least connections
RLB	Randomised load balancing

[1]School of Computing Science and Engineering, Galgotias University, Greater Noida, UP, India. [2]Department of CSE, Ajay Kumar Garg Engineering College, Ghaziabad, UP, India. [3]Department of CEA, GLA, Mathura, UP, India. [4]Arba Minch University, Arba Minch, Ethiopia. [5]Department of Mechatronics, Faculty of Engineering, Ain Shams University, Cairo 11566, Egypt. [6]Department of Industrial Engineering, College of Engineering, King Saud University, P.O. Box 800, 11421 Riyadh, Saudi Arabia. [7]Department of Mathematics, College of Science and Arts, Muhayil, King Khalid University, 61413 Abha, Saudi Arabia. [8]Department of Computer Science, College of Computing and Information Technology, Shaqra University, 11961 Shaqra, Saudi Arabia. [9]Department of Computer Science, Faculty of Computer and Information Sciences, Ain Shams University, Cairo 11566, Egypt. ✉email: saritasimaiya@gmail.com; abdelaziz@su.edu.sa

PSO	Particle swarm optimization
TET	Total execution time
BD	Balance degree
QoS	Quality of services
SLA	Service level agreement
RLB	Randomised load balancing
PSO	Particle swarm optimization
SA	Simulated annealing
GA-SA	Hybrid genetic algorithm and simulated annealing
ABC	Artificial bee colony
FA	Firefly algorithm
GA-PSO	Hybrid genetic algorithm and particle swarm optimization
FL-LB	Fuzzy logic-based load balancing
IoT	Internet of things
RU	Resource utilization
MIPS	Millions of instructions per second
VM	Virtual machine

Cloud computing has transformed the IT industry, influencing how businesses and individuals access and use computing resources. It offers scalable and flexible solutions based on a "pay-per-use" model, which improves resource utilisation across multiple applications. The accessibility of cloud services has resulted in a significant increase in demand. To support the increasing number of users and workloads, cloud systems must implement more efficient load balancing and resource allocation strategies. Cloud computing significantly improves the efficient implementation of business and scientific operations in a variety of industries. Cloud providers primarily provide SaaS (Software as a Service), IaaS (Infrastructure as a Service), and PaaS[1]. These services are tailored to address the varied requirements of clients across different service tiers. Although cloud services like computing, storage, and networking may fulfil analogous roles, they vary in non-functional attributes referred to as QoS (Quality of Service) parameters. The primary factors influencing service quality are response time, cost, availability, energy consumption, and resource utilisation[2].

As cloud systems grow more complex, effective load balancing and resource management become increasingly vital. A plethora of researchers have examined heuristic algorithms and machine learning techniques, often focusing on single-objective optimisation models. However, these conventional methods fail to accommodate the dynamic and continuously evolving characteristics of cloud environments[3]. Traditional load-balancing techniques encounter numerous limitations, including inadequate scalability, sluggish convergence rates, and a failure to tackle multiple conflicting objectives simultaneously. A significant drawback of these methods is their failure to adjust to the rapidly evolving conditions typical of contemporary cloud computing workloads[4]. Traditional methods, particularly heuristic algorithms, are designed for more minor, more straightforward problems and become insufficient as cloud infrastructures grow in size and complexity.

Furthermore, single-objective optimisation techniques typically emphasise maximising resource utilisation or minimising response time while frequently overlooking other critical factors such as energy consumption and system reliability. This limited concentration may result in subpar performance, ineffective resource utilisation, and increased operational expenses[5]. With the increasing dynamism and scale of cloud systems, there is an imperative demand for multi-objective optimisation techniques capable of concurrently balancing competing objectives, including the reduction of response times, optimisation of resource utilisation, and minimisation of costs.

In cloud computing environments, these optimisation goals frequently conflict. For example, decreasing response time may necessitate the assignment of additional resources to tasks, potentially increasing energy consumption and operational costs. Alternatively, emphasising energy conservation may result in longer response times and lower service quality. Achieving an optimal balance between these competing objectives is difficult, as advances in one domain frequently impede progress in another. An effective cloud optimisation strategy must carefully balance multiple objectives in order to effectively manage all system components, such as performance, cost, energy consumption, and reliability.

This study seeks to tackle these challenges by introducing a hybrid optimisation approach that integrates ACO and WWO. The proposed method enhances the management of multi-objective cloud computing environments by utilising ACO's effective local search for task scheduling and WWO's global search capabilities for resource allocation. The objective is to enhance response times, optimise resource allocation, and decrease operational expenses, thereby delivering a more scalable and efficient solution for contemporary cloud systems.

Problem statement and objectives

The rapid expansion of cloud computing introduces increased complexity in the management of resources and services. As user numbers rise and workloads diversify, effective load balancing and resource allocation are essential for maintaining continuous and high-quality service. Given the dynamic nature of cloud environments, where demands can change rapidly, it's vital to have advanced optimisation techniques that can adapt and maintain system performance across all conditions.

A principal challenge in cloud computing is combining diverse, often conflicting optimisation objectives. Optimising resource utilisation is essential for improving system efficiency; however, it may lead to increased response times or higher operational expenses. Conversely, decreasing response times may require the allocation of supplementary resources to particular tasks, potentially adversely impacting overall resource efficiency and increasing energy consumption. The main challenge is to develop an optimisation method that balances conflicting goals without compromising any aspect, ensuring optimal overall performance. To tackle this, the study proposes a hybrid approach combining ACO and WWO.

The goal is to develop a multi-objective optimisation model that reduces response times, maximises resource utilisation, and cuts operational costs. This hybrid approach blends the local search strengths of Ant Colony Optimization for task scheduling with the global search capabilities of Whale Optimization for resource allocation. Together, these methods offer a balanced, comprehensive solution for cloud resource management. A few of the primary goals of this research are as follows:

- *Multi-objective optimization:* Create a model that effectively balances crucial goals like reducing response times, optimising resource usage, and lowering operational costs. The goal is to achieve the best possible overall performance while simultaneously addressing all of these goals.
- *Efficient load balancing and resource allocation:* The proposed method combines ACO's ability to search locally with WWO's global exploration capabilities to improve task scheduling and resource allocation. This balanced approach ensures that the system runs more efficiently, even when demands and workloads change.
- *Cost and performance efficiency:* This study focuses on reducing operational expenses while enhancing resource efficiency and response times. The hybrid approach is designed to improve system performance while maintaining service quality, allowing it to be adaptable to the ever-evolving requirements of contemporary cloud environments.

This research seeks to tackle the challenges of managing cloud resources by introducing a hybrid optimisation method. By combining the strengths of ACO and WWO, the proposed approach aims to balance multiple competing objectives, offering a more efficient and scalable solution for today's complex cloud systems.

Motivation of the research

This research is motivated by the increasing complexity and demand in cloud computing environments, where traditional optimisation methods do not meet performance, cost, and scalability objectives. The rapid evolution of cloud applications and the growing diversity of workloads pose significant challenges for conventional load balancing and resource allocation methods, which often fail to adapt effectively in real time. The principal issue in cloud computing is the inefficacy and suboptimal performance of current algorithms when utilised for dynamic and varied workloads. Conventional methods frequently encounter challenges, including elevated latency, inadequate scalability, and an inability to optimise multiple objectives concurrently. As cloud environments grow in scale and complexity, these shortcomings become more apparent, leading to performance bottlenecks, increased operational costs, and diminished user satisfaction[6].

This research seeks to create a novel hybrid optimisation method that targets substantial deficiencies. We aim to create an innovative methodology by combining WWO and ACO to effectively tackle multi-objective optimisation problems in cloud computing environments. The hybrid methods employs ACO's strong local search capabilities for task scheduling and WWO's global exploration benefits for efficient resource allocation. This combination seeks to augment system efficiency, diminish response times, optimise resource allocation, and lower operational expenses, thereby enhancing the user experience. The research aims to advance cloud computing by delivering a solution that improves performance while providing a scalable and adaptable model to tackle the increasing complexity of contemporary cloud systems. The proposed hybrid model aims to enhance the optimisation of cloud operations while improving system sustainability and reliability.

Key contributions

Conventional workflow scheduling research primarily considers optimisation under time or cost constraints, ignoring energy consumption[6,7]. To address the key challenges of Single-objective models in a dynamic cloud workflow scheduling environment, this research presents an efficient multi-objective hybrid model using the key features of WWO and ACO. The key contribution of this research mainly includes the following.

- *Development of hybrid algorithm*: An innovative hybrid optimisation algorithm combines the strengths of ACO and WWO. This approach enhances cloud load balancing and resource allocation by leveraging the power of ACO's local search and WWO's global search features.
- *Multi-objective optimization*: The proposed hybrid algorithm is based on multi-objective concepts. It simultaneously achieves the minimum response time, maximises resource efficiency, and reduces operational costs. The optimisation method is a balanced solution to achieve diverse performance metrics in dynamic cloud scenarios.
- *Performance validation*: The research validates the hybrid approach with rigorous simulations and comparisons to GA, SMO, and ACO. The proposed optimisation methodology improves user satisfaction and system efficiency, proving its practicality and efficacy.

Structure of the work

The article presents systematic hybrid WWO-ACO cloud computing optimisation research. The introduction emphasises load balancing and resource allocation in dynamic cloud environments. A comprehensive literature review covers existing research and methods. The hybrid optimization algorithm's development and implementation are covered in materials and methods. The hybrid approach is compared to GA, SMO, and ACO in detail through simulations and results. Critical analysis considers system performance, user satisfaction, and operational efficiency. The conclusion highlights key findings, practical implications, and future research directions.

Literature review

Cloud computing has transformed the management and distribution of resources across networks. A primary challenge in cloud computing is effective resource management, especially via load balancing. Load balancing is the technique of evenly distributing workloads among multiple computing resources to ensure optimal resource utilization, reduce response time, and enhance system performance. Effective load balancing is essential for maintaining the performance, scalability, and reliability of cloud systems, especially when handling large and dynamic workloads.

With the advancement of cloud computing, researchers concentrate on creating innovative algorithms to tackle the intricacies of load balancing. Metaheuristic algorithms demonstrate significant potential, offering flexible, near-optimal solutions for dynamic resource allocation in cloud environments. These algorithms are designed to address optimization challenges in cloud computing that go beyond the limitations of traditional methods. This review looks at recent advances in load balancing methods in cloud computing, focusing on metaheuristic algorithms, energy-efficient scheduling, and hybrid optimization strategies.

Review based on cloud computing load balancing techniques

The uniform distribution of work across all available resources is a fundamental component of cloud computing, which is known as load balancing. In an effort to optimize load balancing in cloud environments, Ghafir et al. (2024) presented a novel feedback controller that uses Particle Swarm Optimization (PSO). This method improves the efficiency of workload distribution in cloud infrastructure and saves resources by simplifying the task assignment process to virtual machines[1]. Dhabliya et al. (2024) proposed new rules and methods for improving cloud computing dynamic load balancing. These strategies address the reality that cloud workloads evolve over time by ensuring that resources can adapt and respond to fluctuations in demand[2]. Khan (2024) investigated dynamic load balancing utilizing reinforcement learning-based clustering methodologies and multi-objective task scheduling. This methodology allows the system to adapt to variations in the cloud environment, improving system throughput and resource utilization[3]. Dubey and Mishra (2024) evaluate various load balancing algorithms, highlighting their efficacy and dependability in cloud computing settings. Their research highlights the importance of scalability and robustness in managing diverse workloads[4].

Choudhary and Rajak (2024) proposed an improved min-min heuristic method for workflow scheduling that prioritizes deterministic task allocation. This algorithm improves task completion times and ensures optimal load distribution, resulting in higher computational efficiency[5]. Geetha et al. (2024) investigated hybrid optimization algorithms for load balancing, demonstrating that combining various optimization methods improves resource management, increases energy efficiency, and lowers operational costs[8].

Review based on metaheuristic algorithms in cloud resource management

Metaheuristic algorithms, including Particle Swarm Optimization (PSO), Genetic Algorithms (GA), and Ant Colony Optimization (ACO), play a vital role in addressing intricate resource management challenges in cloud computing. These algorithms are proficient at identifying near-optimal solutions in dynamic, large-scale systems. Forghani et al. (2024) presented the Krill Herd metaheuristic algorithm aimed at load balancing and energy optimization in software-defined networks. This method dynamically modifies cloud resources according to workload variations, guaranteeing load balancing and energy efficiency[9]. Similarly, Li et al. (2024) presented the Tactical Unit Algorithm, a novel metaheuristic technique for optimizing the load distribution in energy systems, such as chiller systems in cloud environments, demonstrating the algorithm's utility in cloud resource management[10].

Singh et al. (2024) utilized a JAYA-based metaheuristic algorithm to enhance workload distribution within a Fog-Cloud ecosystem. This method enhances load balancing while also emphasizing energy efficiency throughout both cloud and edge computing layers[6]. Tiwari et al. (2024) employed a Knapsack-based metaheuristic to enhance edge server placement in 5G networks, effectively balancing computational loads and network capacity[7].

Review based on energy-efficient scheduling in cloud computing

Energy efficiency represents a critical challenge in contemporary cloud computing systems. Data centers consume vast amounts of energy, and efficient scheduling can significantly reduce operational costs and environmental impact. Techniques for scheduling that prioritize energy efficiency focus on reducing energy usage without compromising system performance.

Energy-efficient solutions are increasingly significant in cloud computing, driven by concerns regarding environmental impact and operational expenses. Singhal et al. (2024) employed the Rock Hyrax Optimization algorithm to formulate a solution for energy-efficient load balancing in cloud computing. The algorithm improves resource allocation and minimizes energy usage, making it ideal for large-scale cloud implementations[11]. Priyadarshi (2024) presented a comprehensive analysis of AI and metaheuristic methodologies for energy-efficient routing in wireless sensor networks, applicable to cloud systems as well. His research emphasizes multiple strategies for minimizing energy consumption while maintaining optimal system performance[12]. Rostami et al. (2024) presented a hybrid methodology that integrates Capuchin Search with Inverted Ant Colony Optimization to facilitate energy-efficient task scheduling in heterogeneous cloud environments. Their approach optimizes execution time and energy consumption, guaranteeing the cloud system functions efficiently while reducing energy usage[13].

Hou et al. (2024) examined the utilization of deep reinforcement learning (DRL) for energy-efficient task scheduling in cloud settings. DRL can dynamically modify task schedules to minimize energy consumption while maintaining optimal performance of cloud systems[14]. Khaleel (2024) presented a dynamic job scheduling approach that integrates adaptive chaotic sparrow search optimization with coalitional game theory to attain load balancing and energy efficiency in cloud environments[15].

Review based on hybrid optimization approaches in cloud computing

Hybrid optimization techniques have become popular due to their ability to combine the best features of different optimization methods. These techniques incorporate different metaheuristic algorithms. These approaches make it easier to resolve complex problems that call for balancing multiple goals, such as security, energy efficiency, and performance.

Simaiya et al. (2024) suggested a hybrid approach to load balancing and host utilization forecasting that integrates optimization and deep learning techniques. This model forecasts future workloads, allowing the cloud system to allocate resources proactively and maintain peak performance[16]. Kak et al. (2024) proposed a hybrid metaheuristic method for minimizing energy consumption in cloud systems and optimizing task scheduling, resulting in a balanced solution that improves system performance while lowering operational costs[17]. Behera and Sobhanayak (2024) employed a hybrid Genetic Algorithm—Grey Wolf Optimization (GA-GWO) approach for task scheduling in heterogeneous cloud environments. Their methodology enhances both task completion time and energy efficiency, rendering it appropriate for resource-diverse environments[18]. Verma (2024) presented a hybrid model that integrates Seagull Optimization and Black Widow Optimization for task scheduling in cloud environments. This hybrid methodology tackles performance enhancement and security, illustrating how optimization algorithms can fulfill the dual requirements of efficiency and safety[19].

The literature addresses various novel approaches to load balancing and resource management in cloud computing. Metaheuristic algorithms such as PSO, GA, ACO, and RL provide effective solutions to complex optimization problems. Hybrid optimization techniques show promise for improving performance while addressing important goals like energy efficiency, security, and scalability. As cloud systems evolve, the creation of smarter, more adaptive algorithms will be critical to maintaining efficient and high-performing clouds. Table 1 presents a comparative analysis of various load-balancing methods based on various parameters.

Problem formulation and modeling

In modern cloud computing environments, resource utilization across data centres is frequently inefficient, with physical machines (PMs) typically operating at 15% to 30% capacity. This underutilization results in a large number of idle PMs, which account for approximately 65% of total energy consumption during peak periods. To enhance energy efficiency and reduce operational costs, it is critical to develop methods that optimize resource utilization and minimize energy waste. Load balancing and effective resource allocation play a crucial role in improving the overall performance and cost-effectiveness of cloud systems. The presented research introduces a multi-objective hybrid optimization model that rectifies the limitations of traditional single-objective methods in cloud workflow scheduling. Conventional methods predominantly emphasize the optimization of time or cost, often overlooking energy consumption.

The proposed model amalgamates WWO and ACO to provide a balanced solution for multiple objectives, including performance enhancement, cost reduction, and increased energy efficiency for cloud service providers (CSPs) and cloud customers (CCs). The model aims to optimize resource utilization, reduce idle physical machines, and efficiently allocate workloads, thereby preventing system overload and improving performance. It lowers operational costs while adapting to changing cloud environments, ensuring scalability and sustainability. Focusing on these key objectives improves service delivery, reduces energy consumption, and increases cloud system efficiency and customer satisfaction.

Multi-objective optimization (MOO) model

Different stakeholders in cloud computing systems have unique goals that must be coordinated to ensure smooth operation. The cloud comprises two primary components: Cloud customers (CCs) and Cloud service providers (CSPs). Cloud service providers provide resources to customers, who can then submit their tasks for processing. Service providers maximize resource utilization to boost profits, while consumers prioritize application performance. As a result, the goals can be classified into two primary categories: those that focus on the Customer and those that focus on the cloud service provider[22].

Let the set of Tasks be T, Set of Virtual Machines VMs. Here $T = \{T_1, T_2, T_3, T_{4,\ldots\ldots}T_n\}$, here T_i Represents the task set, i shows the task number which needs to be processed. For each of the task T_i , an execution time $ET(T_i)$. Similarly, a sect of VMs can be defined as $VM = \{VM_1, VM_2, VM_3, VM_{4,\ldots\ldots}VM_m\}$.

- *Pareto-dominance and fitness evaluation*: To handle the conflicting objectives of the stakeholders, the multi-objective optimization (MOO) model considers several factors such as response time, energy consumption, execution cost, and load balancing. The outcomes produced by the hybrid WWO-ACO algorithm are assessed according to these objectives. For two candidate solutions, *S*1 and *S*2, *S*1 is considered to Pareto dominate *S*2 if *S*1 is at least equivalent to *S*2 in all objectives and superior in at least one. The Pareto dominance process guarantees that the optimization method approaches an optimal set of trade-offs.
- *Selection of Pareto-optimal solutions*: We monitor all candidate solutions generated by the hybrid WWO-ACO algorithm during the optimization process. The Pareto front, which represents the optimal set of trade-offs, is determined by iteratively evaluating the solutions through Pareto dominance.
- The ultimate solution set comprises the Pareto-optimal solutions, indicating that no alternative solution surpasses them across all objectives. These solutions exemplify optimal compromises between competing objectives, providing the decision-maker with a spectrum of options from which the most appropriate solution can be chosen according to particular preferences.

Multi-objectives based on customer

Consumers of cloud services are primarily concerned about the cost-effectiveness and performance of task completion[16].

Method	Response time	Waiting time	Energy consumption	Degree of load balancing	Cost of execution	Scheduling length	Scalability	Fault tolerance	Complexity	References
Whale Optimization Algorithm	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	O(n^2) (quadratic due to iterative population evaluation)	[1]
Water Wave Optimization	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	O(n^2) (similar to WOA, with moderate iterative steps)	[2]
Genetic Algorithm	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	O(n × m) (population size × number of generations)	[3]
Particle Swarm Optimization	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	O(n × m) (population size × number of iterations)	[4]
Ant Colony Optimization	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	O(n × m) (number of ants × number of iterations)	[5]
Adaptive Genetic Whale Optimization	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	O(n × m) (Genetic operations plus whale optimization)	[8]
Inverted Ant Colony Optimization	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	O(n × m) (number of ants × number of iterations, less complex)	[9]
Fuzzy-Based PSO	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	O(n × m) (iterative updates of particle positions)	[10]
Ripple-Induced Whale Optimization	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	O(n^2) (due to ripple effects and evaluation complexity)	[6]
Grasshopper Optimization Algorithm	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	O(n × m) (population × number of iterations)	[12]
Meta-Heuristic Approaches for Microgrid	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	O(n × m) (depending on the nature of metaheuristics)	[14]
Load balancing via intelligent PSO	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	O(n × m) (particle updates per iteration)	[20]
Dynamic Load Balancing via Optimized RL-Based Clustering	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	O(n^2) (RL-based updates + clustering evaluation)	[15]
Modified min-min heuristic	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	O(n × m) (iterative heuristic-based evaluation)	[16]
Optimal load balancing via hybrid optimization	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	O(n^3) (complexity due to hybrid optimization processes)	[17]
Dynamic optimization via utilizing the Krill Herd meta-heuristic algorithm	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	O(n × m) (meta-heuristic iteration-based, high overhead)	[18]
Tactical Unit Algorithm	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	O(n^2) (time complexity increases with more tactical units)	[21]
Optimising workload distribution via a JAYA-based meta-heuristic	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	O(n × m) (iterative updates, with moderate computational cost)	[19]
Hybrid WWO-ACO	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	O(n^2) (a hybrid of WWO and ACO, the computational cost is moderate)	Proposed

Table 1. Comparative analysis of various load-balancing methods based on various parameters.

Based on the length of scheduling Schedule length is the maximum time required to complete all assigned assignments and the most recent processed VMs (Virtual Machines). This metric is crucial for evaluating the scheduler's performance. A minimal schedule length result demonstrates an effective scheduling technique that assigns assignments to the correct resources[17]. Schedule length Sch_{length}. This can be calculated by Eq. (1).

$$Sch_{length} = \max\left[\sum_{i=1}^{n} ET(TL, VM_i)\right] \tag{1}$$

where ET represents execution time, TL Represent the task length, i represents the task number that will be executed, which can be calculated by Eq. (2), Cl_{length} represents cloud let length, TI represents the execution of the total no. of instructions (MIPS), and it can be calculated by Eq. (3). Equation (4) represents the calculation for the single-machine execution $TI_i(VM_i), hereVMP_{Power}$ represents the VM capacity, and VM_{Tcores} represents the total number of cores in the VM.

$$ET\left(Cl_{length,}VM_{j}\right)=\frac{Cl_{length}}{\sum TL(VM_{i})} \tag{2}$$

$$\sum TI(VM_{i})=TI_{1}(VM_{1})+TI_{2}VM_{2}+\ldots\ldots\ldots\ldots\ldots+TI_{n}VM_{n} \tag{3}$$

$$TI_{i}(VM_{i})=[VMP_{Power}*VM_{Tcores}] \tag{4}$$

In Eq. (4) $\mathrm{TI_i}(\mathrm{VM_i})$ represents the total number of instructions for the ITH task and the virtual machine; it is measured in MIPS (Millions of Instructions Per Second), and the $\mathrm{VMP_{Power}}$ is also measured in MIPS.

Task execution cost (TEC) Without being aware of the intricacies of the system infrastructure, cloud users in the cloud environment hand over their autonomous tasks to the service provider for processing. These tasks have differences regarding prerequisites, such as resource requirements and task duration[18]. The total expense of processing a user's application is the TEC. Although it's usually the most quantifiable metric available today, it's crucial to express the cost in terms of the available resources. Given that the user wants to shorten schedules and save costs. The execution cost of this arrangement scheme can be calculated by Eq. (5), Where TEC is the task execution cost, TET is the task execution time, i is the task number, $\mathrm{TE_i}$ is the ith task price.

$$TEC=[TET_{i}*TE_{i}] \tag{5}$$

Service provider-based objectives
In order to reduce operating expenses and improve service dependability, service providers concentrate on effective resource management.

Energy consumption The utilisation of CPU, network connections, and storage equipment contributes to energy utilization in data centres. The CPU uses more energy than the other system resources. A virtual machine's energy consumption can be separated into idle and active categories. The two VM states are considered when determining the total amount of energy used. For the cloud infrastructure to be sustainable and economical, total energy consumption is presented in Eq. (6). Here, subscript base, cap n , end base , sub i. represents the energy consumption in kilowatt-hours (kg) by resource i, and subscript base, can Pc , end base, sub i. represents the power consumption rate in watts (W), for a particular resource i.Pc_i represents the power consumption rate for an individual or a particular resource i.

$$MinEnergy_{Consumption}=\sum(En_{i}*Pc_{i}) \tag{6}$$

Resource consumption (RC) Providers of cloud computing services must effectively control their computing resources to guarantee long-term survival and maximize financial returns. Perfect operation of data centres, which mostly rely on CPUs, RAM, storage, and network bandwidth, depends on effective resource management. Effective resource management aims to optimize occupancy rates, minimize waste, and improve general efficiency.

Cloud service providers constantly search for the most effective ways to use their resources to maximize their return on investment. This calls for effective processing and storage capacity management, the best use of current resources, and low idle times, which reduces needless costs and minimizes energy consumption that does not help generate income, which depends on effective resource management. Cloud service providers can boost productivity, cut costs, and improve profitability through efficient resource utilization. Efficient resource management is crucial for the smooth operation of the data centre, helping the provider achieve their business objectives and ensuring overall success.

Materials and methods

This section thoroughly explains the proposed hybrid model, covering its conceptual framework, operational mechanisms, and performance evaluation methodologies. The hybrid approach, which combines the strengths of WWO and ACO, is thoroughly examined to showcase its effectiveness in cloud load balancing and resource allocation. In addition, this section explains the main performance metrics used to assess the model's effectiveness, ensuring a thorough comprehension of its efficiency and influence. This comprehensive analysis highlights the model's potential to bring about significant enhancements in the management of cloud resources.

Proposed WWO-ACO hybrid model

Combining the features of WWO and ACO, the proposed hybrid model seeks to solve the pressing issues of load balancing and resource allocation in cloud computing environments. This creative approach is painstakingly built to strike a perfect balance between several goals, including optimizing resource efficiency, minimizing energy consumption, lowering running costs, and so minimizing response times.

Figure 1 presents the architecture of the proposed hybrid model. In the proposed model, starting with a population of solutions, each reflecting several approaches for load distribution and resource allocation, the process starts with WWO looking extensively worldwide to find places with great potential for development. ACO then uses ant foraging behaviour to search locally, improving solutions in these interesting domains. During optimization, the model effectively manages different objectives by continuously adjusting weights using real-time feedback. This enables the model to adapt to changing conditions quickly.

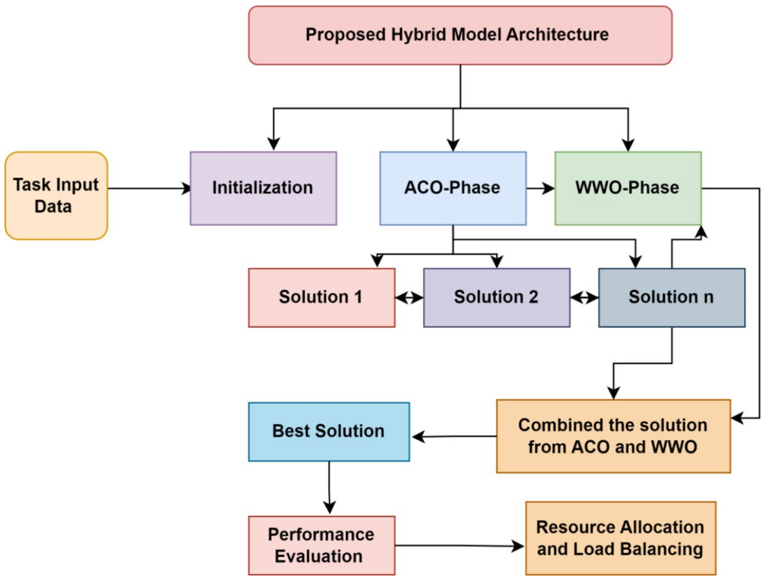

Fig. 1. Architecture of Proposed Hybrid Model.

Effective load balancing and resource allocation are achieved by selecting optimal configurations and parameters. Performance comparisons consider response time, resource utilization, energy efficiency, and operating costs. This process repeats until specific goals are met, such as optimization or a set number of iterations. By combining ACO's local search efficiency with WWO's global search capability, the model guarantees optimal resource utilization, cost reduction, energy efficiency, and improved system performance. This holistic approach promotes the creation of more sustainable and efficient cloud computing environments. The following sections provide a thorough explanation of how the proposed model operates.

ACO working

The ACO algorithm, inspired by ants' foraging behaviour, is based on a probabilistic approach. It is utilized to discover approximate solutions for complex optimisation problems. Algorithm 1 illustrates the comprehensive functioning of ACO[23].

```
Step 1 Initialization Phase
1.1 Initialization of variables and parameters:
  - Set the number of ants m
  - Pheromone evaporation rate φ
  - Pheromone influence parameter$\sigma$
  - Heuristic influence parameter $\beta_1$
  - Initial pheromone levels $T_{ij}$ $(0)$ for all the respective edges i and j.
1.2 Task and Resource Definitions
  - Define the set of tasks $T = (T_1, \dots \dots \dots . T_n)$
  - Define the set of resources $R = (R_1, \dots \dots \dots . R_m)$
  - Define the initial pheromone level $T_{ij}$Between tasks and resources.
Step 2. The phase of Solution Design
 Each ant builds a solution by iteratively choosing resources for tasks based on pheromone levels and heuristic information.
2. 1 Probability Calculation:
  - Ants create solutions using probability, knowledge from pheromone levels, and heuristics to select resources. The Equation 7 is used to measure the probability level$PB_{ij}$.
      $PB_{ij}(t) = \frac{(PL_{ij}(t))^{\varpi} * (HV_{ij})^{\beta 1}}{\sum_{k \varepsilon N_i} (PL_{ik}(t))^{\varpi} * (HV_{ik})^{\beta 1}}$      (7)
Where $(PL_{ij}(t))$ represents pheromone level, $(HV_{ij}$ heuristic value, $\varpi$and $\beta_1$ represents parameters controlling
2.2. Ant Solution Construction:
  - Each constructs a complete solution by probabilistically selecting resources for each task based on $PB_{ij}(t)$.
Step 3. Pheromone Update Phase
3.1. Local Pheromone Update:
  - Once a solution has been constructed, each ant individually maintains the local pheromone scales by using equation 8.
      $T_{ij}(t) = (1 - \varphi) * T_{ij}(t) + \varpi * T_{ij}(0)$      (8)
    Where: $\varphi$ represents the pheromone evaporation rate and $T_{ij}$ $(0)$ is the initial pheromone level.
3.2. Global Pheromone Update:
  - Once every ant has built their solution, Equation 9 updates the global pheromone based on how well-built the solutions are.
      $T_{ij}(t+1) = (1 - \varphi) * T_{ij}(t) + \Delta T_{ij}$      (9)
   Where:
   - $\Delta T_{ij}$ It represents the amount of pheromone deposited, which can be calculated by Equation 10.
      $\Delta T_{ij} = \sum_{k=1}^{m} (\Delta T_{ij}{}^{k})$      (10)
   where:
   -$(\Delta T_{ij}{}^{k})$ represents pheromone for kth ants
Step 4. Evaluation Phase
4.1 Fitness Evaluation:
Evaluate the performance using the objective model, Equation 11.
   $Mult_{solution} = W_1 RT(solution) + W_2 RC(solution) + W_3 OC(solution) + W_4 EC(solution)$      (11)

Where $RT$ represents response time, $OC$ represents operation cost, $RC$ represents resource consumption, and $EC$ for energy consumption.
4.2 Selection of Best Solution:
Again, apply the best fit $(Best_{solution})$ finding by using local and global phenomena solutions, Equation 12.
      $Best_{solution} = Arg_{\, solution\, \varepsilon (ACO, WWO)}\, Best\ (Fit_{solution})$      (12)
```

Algorithm 1. ACO Algorithm

The ACO algorithm efficiently explores the search space using heuristic information and probabilistic decisions based on pheromones. It balances exploration and exploitation, making it possible to find excellent solutions for resource allocation and cloud load balancing.

WWO working

A WWO algorithm is based on the movement and interaction of water waves; it is an optimisation algorithm that is highly inspired by nature. Using the concepts of wave propagation, refraction, and breaking concepts, this algorithm simulates waves' motion in lakes, rivers, and oceans to find the best solutions within a specified search space. The WWO algorithm consists of the following main steps: selection, wave propagation, wave refraction, wave breaking, and initialization. Algorithm 2 and Pseudo code 1 thoroughly summarise every step of WWO[24].

Step 1. *Initialization Phase*

1.1 The population of waves is initialized at this phase using the ACO phase solutions. In the search space, each wave denotes a potential solution.

1.1.1 Wave Position: A wave's position (x_i) in the search space defines it.

1.1.2 Wave Height: Each wave is accompanied by a corresponding height (h_i), representing the solution's fitness level. Greater wave heights indicate superior solutions.

Step 2. *Wave Propagation Phase*

2.1 During this stage, every wave moves and spreads to a different location, as presented by Equation 13.

$$X_{new}^{i} = [X_i + (\varpi \times h_i) \times rand(n)] \qquad (13)$$

Where:

-ϖ represents a constant scaling factor, h_i represents wave height and $rand(n)$ represents a random number selected from a normal distribution.

The new position is calculated by adding the displacement vector across the present state of the wave.

The displacement vector usually represents a stochastic vector multiplied by the wave height.

Step 3. *Wave Refraction Phase*

3.1 The algorithm carries out wave refraction following wave propagation.

A reflection technique is employed to correct a wave's location if it exceeds the search space's bounds. It can be calculated by Equation 14.

$$X_i^{Refraected} = (X_i + \beta_2) \times \left(X_{Boundary} + X_i\right) \qquad (14)$$

Where:

- β_2 represents a refraction coefficient and $X_{Boundary}$ represents a boundary for a particular search space.

Step 4. *Wave Breaking Phase*

4.1 Wave breaking is how new waves replace old waves with poor fitness (tiny wave heights). By taking this action, early convergence is avoided, and population diversity is preserved.

4.2 Waves shorter than a predetermined threshold are broken and replaced by freshly generated waves at random.

Step 5. *Final Selection Phase*

5.1 This phase determines which waves are the best to form the next generation after propagating, refracting, and breaking.

5.2 The waves' fitness values serve as the basis for selection.

5.2.1 Each wave's fitness is assessed using a predetermined objective function.

5.2.2 More fit Waves (have better answers) have a higher chance of being chosen for the following generation.

Algorithm 2. Water Wave Optimization Working

WWO is a robust and adaptable optimization algorithm that efficiently explores and exploits the search space by imitating natural wave behaviours. Utilizing wave propagation, refraction, and breaking mechanisms, WWO can identify superior solutions for challenging optimization issues, such as multi-objective models and cloud load balancing. Pseudo Code 1 presents the complete code of the WWO method[25].

```
Pseudo Code 1: Water Wave Optimization Pseudo Code
1. Initialize a population of waves with random positions and heights
2. Evaluate the fitness of each wave
3. while the stopping criterion is not met, do
4.     for each wave, do
5.         Propagate the wave to a new position
6.         if the new position is out of bounds, then
7.             Refract the wave
8.         end if
9.         Evaluate the fitness of the new position
10.        if new fitness is better than current fitness, then
11.            Update the wave's position and height
12.        else
13.            Apply wave breaking if height is below a threshold
14.        end if
15.    end for
16.    Select the best waves for the next generation
17. end while
18. Return the best solution found
```

Pseudo Code 1. Water Wave Optimization Pseudo Code

WWO-ACO hybrid phase

The complete working of the proposed hybrid model is presented by algorithm 3[26].

Step 1: *Initialization Phase*

Initialize all the necessary parameter variables for WWO and ACO.

Set propagation rates, pheromone levels, wave heights and evaporation rates for WWO and ACO.

Define the Task (T) and Resources (R) $T = \{T_1, \ldots \ldots . T_n\}$ *and* $R = \{R_1, \ldots \ldots . R_m\}$

Step 2: *ACO phase*

2.1 ACO solution creation

Ants create solutions by selecting resources using probability, knowledge from pheromone levels, and heuristics. Equation 15 measures the probability level.PB_{ij}.

$$PB_{ij}(t) = \frac{(PL_{ij}(t))^{\varpi} * (HV_{ij})^{\beta_1}}{\sum_{k \varepsilon N_i} (PL_{ik}(t))^{\varpi} * (HV_{ik})^{\beta_1}} \qquad (15)$$

Where $(PL_{ij}(t))$ *represents pheromone level,* $(HV_{ij}$ *heuristic value,* ϖ*and* β *represents parameters controlling*

2.2 ACO Update Phenomena

In this phase, the Phenomena are updated based on the best solutions, as shown in Equation 16.

$$PD_{ij}(t+1) = (1-\varphi)PD_{ij}(t) + \Delta PD_{ij} \qquad (16)$$

Where PD_{ij} *represents the pheromone deposit,* φ *represents the evaporation rate by Equation 17.*

$$\Delta PD_{ij} = \sum_{k=1}^{m} \frac{Q}{C_m} \qquad (17)$$

Where C_k *represents the cost for a* m*th* *ant solution,* Q *denotes a constant value.*

Step 3. *WWO Phase*

3.1 Initialization of Waves:

Initialize the initial Waves using the best solution from ACO solution buckets by Equation 18.

$$Wh_i(0) = [Fit_{solution_i} + random_{dist}()] \qquad (18)$$

Where $Wh_i(0)$ *represents the height of* i*th* *waves,* $Fit_{solution_i}$ *represents fitness solutions for ith waves and* $random_{dist}()$ *represents a random distribution function.*

3.2 Wave Propagation Phase

This phase utilizes a perturbing process for wave propagation, as presented by Equation 19.

$$Fit_{solution_i}(t+1) = Fit_{solution_i}(t) + \Delta Fit_{solution_i} \qquad (19)$$

Where $\Delta Fit_{solution_i}$ *represents permutation by using the propagation rate and height of a wave.*

3.3 Wave breaking

When waves reach a specific threshold in height, they break and create new waves.

Step 4. *Hybrid WWO-ACO Phase*

4.1 Select the best-fit solution from ACO and WWO solutions buckets.

4.2 Again, apply the best fit $(Best_{solution})$ *finding by using local and global phenomena solutions, Equation 20.*

$$Best_{solution} = Arg_{\,solution\ \varepsilon(ACO,WWO)}\ Best\ (Fit_{solution}) \qquad (20)$$

Step 5. *Performance Evaluation*

5.1 Evaluate the performance using the objective model, Equation 21.

$$Mult_{solution} = W_1RT(solution) + W_2RC(solution) + W_3OC(solution) + W_4EC(solution) \qquad (21)$$

Where RT represents response time, OC represents operation cost, RE represents resource consumption, and EC for energy consumption.

Step 6. *Allocates the Resources allocated based on* $Best_{solution}$ *results allocate the* $Mult_{solution}$ *and achieve load balancing, resource optimization and energy consumption.*

Algorithm 3. Algorithm for proposed hybrid model

Time complexity analysis for WWO-ACO

The time complexity of the hybrid ACO + WWO algorithm is a combination of the individual complexities of both Ant Colony Optimization (ACO) and Water Wave Optimization (WWO). Below is the analysis of the time complexity:

- *Time complexity of ACO*: In ACO, ants explore the solution space iteratively, modifying their paths according to pheromone concentrations. The number of iterations (represented as N_{iter}) and the number of ants (represented as N_{ants}) both affect the time complexity. The time complexity of ACO is quantified using Eq. (22).

$$O\left(N_{ants} \times N_{iter} \times D\right) \tag{22}$$

where D is the dimension of the problem space (i.e., the number of resources or tasks to be allocated).

- *Time complexity of WWO*: The WWO functions by emulating water wave dynamics, with solutions being iteratively refined according to wave propagation. The quantity of waves and iterations affects the complexity. The time complexity of WWO can be calculated by using Eq. (23).

$$O\left(N_{waves} \times N_{iter} \times D\right) \tag{23}$$

- where N_{waves} is the number of wavefronts and N_{iter} is the number of iterations.*Time complexity of hybrid WWO-ACO*: The hybrid algorithm's overall time complexity is determined by the time complexities of both ACO and WWO methods. In practice, the two algorithms operate concurrently or sequentially, and the overall time complexity is generally the aggregate of the individual complexities. The time complexity of the proposed WWO-ACO can be measured by using Eq. (24).

$$O((N_{ants} \times N_{iter} \times D) + (N_{waves} \times N_{iter} \times D)) \tag{24}$$

The parameters N_{ants}, N_{iter} and N_{waves} are adjustable and contingent upon the specific problem and the dimensions of the cloud environment. Adjusting these parameters allows the algorithm to optimize the trade-off between computational efficiency and solution quality.

Dataset

The proposed and existing models are implemented using a Cloud-sim simulator. The proposed WWO-ACO hybrid algorithm's performance is measured using artificial and real-time workload traces. The artificial workload is created using a consistent distribution, which ensures that large, medium, and small workloads are all presented identically. This systematic strategy helps to provide a comprehensive assessment of how well the algorithm performs across different job sizes. A real-time HPC2N (North High-Performance Computing Center)[27] log has been used to check the algorithm efficiency. The HPC2N log serves as a widely recognised and utilised benchmark for evaluating the efficient functioning of systems with distributed components. It serves as a genuine and rigorous testing environment for the proposed algorithm, ensuring reliable results that can be applied to real-world scenarios. Table 2 presents the dataset details.

Simulation and performance analysis

In contrast to existing algorithms in cloud computing environments, this section investigates the performance and simulation of the proposed hybrid WWO-ACO model. We provide a comprehensive analysis that contrasts our hybrid approach with traditional algorithms such as ACO, SMO, GA, and WWO. The comprehensive performance evaluation covers a wide range of scenarios in cloud environments with multiple objectives. It emphasizes critical metrics, including the duration of task scheduling, the cost of task execution, energy consumption, and resource utilization. This comprehensive analysis emphasizes the benefits of the hybrid model and pinpoints areas that require improvement, thereby enabling a clear understanding of its efficiency and effectiveness[28].

Task ids	Task numbers	Task length (MI)	Task properties
1–10	1–10	12,000–120,000	Continuous and free
11–20	11–20	130,000–240,000	Continuous and free
21–30	21–30	250,000–360,000	Continuous and free
31–40	31–40	370,000–480,000	Continuous and free
41–50	41–50	490,000–600,000	Continuous and free
51–60	51–60	610,000–720,000	Continuous and free
61–70	61–70	730,000–840,000	Continuous and free
71–80	71–80	850,000–960,000	Continuous and free
81–90	81–90	970,000–1,080,000	Continuous and free
91–100	91–100	1,090,000–1,200,000	Continuous and free

Table 2. Complete description of the task.

Cloud entity	Characteristic	Value
Data center	Number of data centers	3
	Number of hosts	10
	Number of users	50
Host	Storage capacity	5 TB
	RAM	32 GB
	Bandwidth (BW)	100 GB/s
	Shared policy	Space shared with dynamic allocation
	CPU cores	16 Cores
	Network latency	5 ms
	Power consumption	1 kW
	Virtualization technology	KVM

Table 3. Comparative analysis for Datacenter parameters and Host Configuration.

Characteristic	Value	Description
Number of VMs	50, 100, 150, 200	Total number of virtual machines (VMs) deployed
MIPS	500–1500	Millions of instructions per Second, indicating processing power
Bandwidth (BW)	0.5 Gb/s	Network bandwidth available per VM
VMM	Xen	Virtual machine monitor used for virtualization
Size	100 GB	Storage size allocated to each VM

Table 4. Details Description of VM properties.

Simulation details

The simulation guarantees correct and fast results, utilizing a high-performance computing configuration. Besides 32 GB of RAM, the system boasts an Intel Core i7-11700K CPU running at 3.60 GHz. 64-bit Windows 11 is its running operating system. The cloud environment simulation takes advantage of the most recent iteration of Clouds Plus (CloudSim Plus 6.2.2). Designed especially for cloud computing research, CloudSim Plus is a relatively sophisticated and flexible toolkit. It offers better performance and capabilities than its last versions[29]. Table 3 presents a Comparative analysis of Datacenter parameters and Host Configuration.

Table 3 provides a comprehensive overview of the parameters for the data centre and the host configuration details. We have established three data centres containing ten hosts to accommodate fifty users within our optimised data centre and host configuration. With 32 GB of RAM and 5 TB of storage space, each host is carefully constructed to provide ample resources for running demanding applications at lightning-fast speeds. With a 100 GB/s capacity, the network bandwidth is notable and facilitates quick data flow. Currently, the policy is "Space Shared with Dynamic Allocation," which allows effective and flexible demand-based use of storage resources based on real-time needs.

With sixteen CPU cores, each host ensures sufficient processing capability for concurrent tasks and low network latency of just five milliseconds, thus lowering data transfer delays. With 1 kW power consumption per host, energy efficiency optimisation is the top priority. Moreover, KVM (Kernel-based Virtual Machine) technology guarantees efficient virtualisation and resource allocation. This arrangement makes a harmonic and quite efficient cloud infrastructure possible.

The specifications for the VMs in our cloud setup are listed in Table 4. To accommodate varying computational requirements, we have deployed VMs in batches of 50, 100, 150, and 200. The processing power of each VM ranges from 500 to 1500 MIPS (Millions of Instructions Per Second), so they can effectively handle a wide range of workloads. Data transfer is fast and seamless because each virtual machine has 0.5 Gb/s network bandwidth. Xen is our Virtual Machine Monitor due to its reliable virtualization features. Each 100 GB VM has enough storage for most applications and data. This configuration makes our virtual environment flexible, effective, and able to meet many user needs[30].

Table 5 displays the costs of VM instances ranging from $0.25 per hour for 8 GB of memory and 100 GB of storage, including four cores, to $2.50 per hour with 80 GB of memory, 3200 GB of storage, and 40 cores. Intermediate options offer versatility for different applications and workloads, guaranteeing economical and effective virtual machine deployment[31–33].

Table 6 offers a detailed breakdown of the parameters for the hybrid WWO-ACO algorithm as well as other popular algorithms like GA, SMO, WWO, and ACO. The hybrid algorithm configuration comprises 100 population sizes, 200 iterations, 10 maximum wave heights, 0.1 initial pheromone value, and 0.001 convergence threshold. The population of GA is 100, with 200 iterations, a crossover rate of 0.8, and a mutation rate of 0.01. SMO includes a maximum global learning threshold of 40, a local threshold of 50, and 100 generations. The parameters of WWO and ACO include waves, iterations, and pheromone levels[33–35].

Memory (GB)	Storage (GB)	Cores	Price ($/Hour)	Description
8	100	4	0.25	Suitable for lightweight applications and testing
16	200	8	0.50	Ideal for medium-sized applications and databases
32	400	16	1.00	Perfect for larger applications and data processing tasks
48	800	24	1.50	Excellent for high-performance computing and intensive workloads
64	1600	32	2.00	Optimal for very large databases and complex simulations
80	3200	40	2.50	Best for extensive data analysis and heavy computational tasks

Table 5. Details for VMs instance Cost in $

Algorithm	Parameter	Value	Description
Proposed hybrid WWO-ACO	Population size	100	Size of the population for iterations
	Max iterations	200	Maximum number of iterations for convergence
	Initial pheromone	0.1	Starting amount of pheromone in the algorithm
	Max wave height	10	Maximum height of the waves
	Convergence threshold	0.001	The threshold for determining convergence
GA	Population size	100	Size of the population for the genetic algorithm
	Max iterations	200	Maximum number of iterations for convergence
	Crossover rate	0.8	The rate at which crossover occurs
	Mutation rate	0.01	Mutation Rate
SMO	Global learning limit (GLL)	45	Shows the GLL limit
	Local learning limit (L)	55	Shows the LLL limit
	Max generations (GN)	100	Shows the GN count
	Probability range (pr)	[0.1,0.4]	Shows the pr count
WWO	Number of waves	50	Shows the wave count
	Max iterations	200	Shows the iteration count
	Initial wavelength	0.8	Shows the wave-length count
ACO	Pheromone Importance(q)	0.5	Shows q count
	initial pheromone	0.05	Shows the count for the initial stage
	Number of ants	20	Shows ant count
	max iterations	200	Shows the iteration max count

Table 6. Parameters Details for Existing and Proposed Algorithms.

Simulation results

A thorough test of the hybrid WWO-ACO algorithm included a close look at how it performed against well-known approaches like the GA, SMO, WWO, and ACO. We tested these algorithms in five different cloud computing scenarios with multiple goals. The length of the task schedule and the cost of completing the task were two crucial performance indicators. Shorter schedules and lower costs meant that the task was more efficiently completed. We also measured energy use to see how well each algorithm cut down on energy use; lower energy use means a more environmentally friendly approach.

It was possible to determine how efficient resources were by looking at how many were used. Better resource management meant lower values. Lastly, we used the balance degree metric to see how well the costs, energy, and resource management worked together. This way, we could be sure that improvements in one area didn't negatively affect others. The full study showed the hybrid WWO-ACO algorithm's pros and cons compared to the other methods, explaining how well it works and where it might be improved.

First scenario (using task length)

In this case, our primary focus is on the length of the task schedule, that is, the whole time needed to finish all the allocated tasks. This work aims to assess the degree of reduction in this duration by the hybrid WWO-ACO algorithm, improving the system's speed and efficiency. We compared this hybrid approach with GA, SMO, WWO, and ACO using different task distributions to evaluate its effectiveness. The results demonstrate the hybrid algorithm's capacity to decrease task completion time and enhance efficiency and velocity.

Table 7 compares simulation results for several well-known techniques, i.e., GA, SMO, WWO, and ACO, alongside the newly proposed WWO-ACO hybrid method. This comparison covers various configurations of virtual machines and tasks. The focus is on average and maximum task schedule lengths, and it is evident that the WWO-ACO hybrid consistently outperforms the traditional methods. The WWO-ACO hybrid produces an average schedule length of 1107.8 s for a configuration comprising 2000 tasks and 150 VMs. This is far faster than the ACO algorithm (1407.32), the SMO algorithm (7578.32), the WWO algorithm (8587.7 s), and the

No. of VMs	No of Task	GA Mean	GA Max	SMO Mean	SMO Max	WWO Mean	WWO Max	ACO Mean	ACO Max	WWO-ACO Mean	WWO-ACO Max
50	500	5094.11	7113.5	3975.73	4617.05	5161.73	6925.92	732.28	749.67	713.77	734.62
	1000	6503.32	7033.64	4941.51	5799.8	6440.99	7647.61	893.36	960.36	859.33	877.17
	1500	7682.66	7913.76	5716.45	6444.33	7255.13	8203.59	1042.03	1055.79	997.37	1015.29
	2000	10,526.19	11,386.62	8967.59	9467.53	10,415.96	11,272.7	1446.25	1451.62	1438.46	1439.66
	2500	13,890.20	13,570.81	12,040.23	11,520.87	11,209.70	14,780.92	1870.82	1970.85	1750.62	1890.10
100	500	3153.33	3716.74	2872.9	3365.45	3573.78	4179.36	544.72	689.34	468.42	502.31
	1000	3841.32	4086.98	3281.22	3641.32	4426.14	4968.39	536.34	544.33	495.76	502.44
	1500	4315.43	4328.62	3699.93	3971.04	5432.68	6602.55	643.48	644.32	639.72	641.63
	2000	6160.13	6229.31	5741.72	6245.82	6980.04	7656.74	781.54	783.23	775.04	776.05
	2500	15,700.24	15,500.56	13,790.15	14,200.3	14,370.7	16,480.85	2740.61	2780.83	1980.31	2784.37
150	500	4809.85	5270.56	4087.42	4570.34	5554.75	7850.65	980.82	1320.23	874.17	858.20
	1250	5870.25	6270.48	4580.52	5090.74	6570.5	8023.55	1070.82	1177.23	910.58	945.68
	1500	6807.16	7208.73	5780.83	5500.27	7532.8	9320.74	1280.13	1303.43	1321.7	1100.27
	2000	8301.3	8705.42	5500.48	6000.93	8587.7	12,150.33	1407.32	1598.72	1107.8	1203.06
	2500	8878.83	9209.31	6874.94	6547.52	9532.9	11,095.27	1687.83	1765.93	1203.7	1360.57
200	500	5307.92	5708.28	4578.43	5230.72	6980.28	75,780.91	1108.25	1200.08	958.97	1080.98
	1000	6378.92	6732.37	5230.18	5540.82	7380.37	8577.45	1201.77	1308.17	1032.86	1125.37
	1500	7378.72	7578.32	5507.57	6078.18	8078.78	9578.45	1447.87	1570.80	1107.20	1298.74
	2000	8347.15	8778.12	6230.96	6507.74	9074.51	10,507.89	1601.50	1774.84	1205.84	1387.88
	2500	9378.23	9778.98	6570.98	7032.87	10,780.32	11,570.23	1890.78	1978.95	1310.24	1432.78

Table 7. Evaluating in terms of various task schedule lengths evaluated in seconds the computing outcomes for the current and proposed scheduled tasks.

GA algorithm (7378.72). Remembering that hybrid technology significantly contributes to optimizing cloud environments is crucial. This is evident in the substantial reduction in the average task execution time.

The WWO-ACO hybrid is also adept at distributing the load, with a scheduled time of up to 1203.06 s. It takes 8301.3 s for the GA algorithm, 8705.42 s for the SMO algorithm, 12,150.33 s for the WWO algorithm, and 1598.72 s for the ACO algorithm. It takes 8301.3 s for the GA algorithm, 8705.42 s for the SMO algorithm, 12,150.33 s for the WWO algorithm, and 1598.72 s for the ACO algorithm. These results show that the hybrid can significantly reduce task time, improving system responsiveness and user satisfaction. The WWO-ACO hybrid combines ACO's local search with WWO's global exploration for robust cloud computing multi-objective optimization. Operational efficiency and performance metrics benefit from this integration.

Second scenario (using task execution cost)

This scenario aims to evaluate the efficacy of the hybrid WWO-ACO algorithm by quantifying its effectiveness by measuring Schedule Length, which pertains to the total time needed to finish all planned operations. This scenario aims to reduce the Schedule Length to improve the system's responsiveness and throughput. A comparative analysis has been conducted on the proposed hybrid WWO-ACO algorithm and various existing load balancing methods, i.e., GA, SMO, WWO, and ACO. We have measured the effectiveness of all these methods in different task distribution scenarios. The outcomes show how effectively the system reduces the time needed for task completion and performance.

Figure 2 compares the costs of executing tasks using existing algorithms versus proposed algorithm-based methods. In cloud computing, these expenditures are evaluated by implementing numerous optimization strategies. The results indicate that the WWO-ACO hybrid technique consistently achieves lower costs for task execution in all tested scenarios, spanning from 50 to 200 VMs and 500 to 2500 tasks, in comparison to traditional methods such as GA, SMO, WWO, and ACO. For instance, at 150 VMs and 2000 tasks, WWO-ACO shows a mean cost of \$8301.3, which is notably lower than GA (\$7378.72), SMO (\$7578.32), WWO (\$8587.7), and ACO (\$1407.32). This emphasises the hybrid method's effectiveness in reducing average task expenses and improving cost-effectiveness in dynamic cloud environments.

Furthermore, a closer examination of the costliest activities exposes WWO-ACO's extraordinary capacity for financial control and workload handling. This shows how strong the system is in properly controlling several execution needs. The suggested hybrid approach offers a sensible way to maximize multi-objective activities in cloud computing. This method maximises efficiency by combining the local search powers of ACO with the worldwide exploration powers of WWO. These results imply a possible improvement in performance criteria and operational efficiencies among several cloud systems.

Third scenario (using energy consumption)

In the third scenario of the experimental analysis, we have calculated the Energy consumption for the proposed hybrid WWO-ACO and existing load balancing algorithms to measure their performance and determine efficacy. In a cloud computing environment, the total amount of energy needed to complete a task is the energy consumption for a task. Less energy consumption is always desirable in the cloud computing environment. It

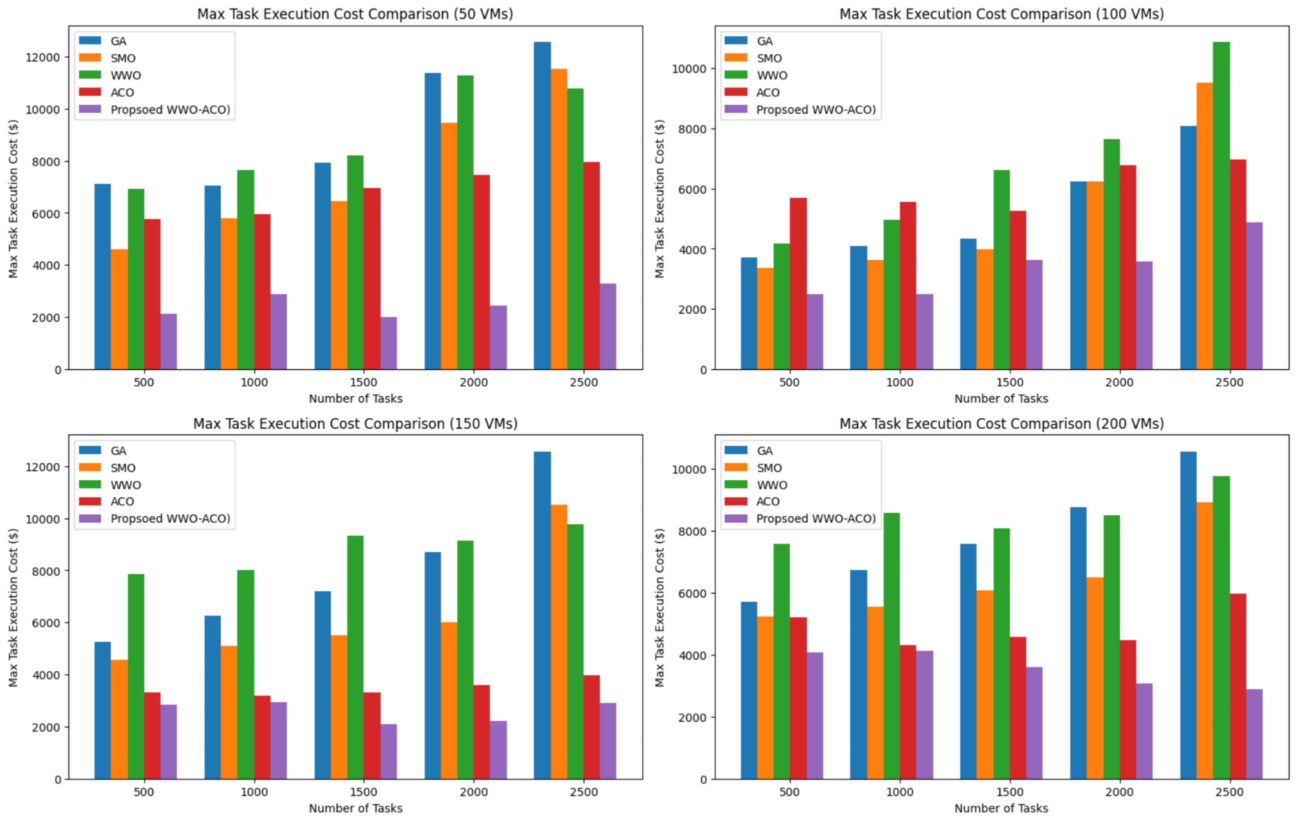

Fig. 2. Analyze and compare the costs of executing tasks using existing algorithms versus proposed algorithm-based methods.

helps increase the system's responsiveness and efficiency and decreases total energy (TE). The effectiveness of the hybrid algorithm in reducing TE in contrast to other existing approaches, like GA, SMO, WWO, and ACO, is assessed through simulations of various workloads and task distributions. The outcomes are examined to show how effectively the suggested strategy maximizes support the execution time, enhancing user satisfaction and system performance.

Figure 3 presents a comparative analysis of existing and proposed algorithm-based energy consumption in kilojoules (KJ). In this regard, we investigated the energy consumption of several Virtual Machine configurations and task loadings. We used several optimization approaches, including the recently proposed WWO-ACO method, ACO, WWO, SMO, and GA. Analyzing both the mean and maximum energy usage enabled the study to concentrate on the effectiveness of these methods. The analysis of the mean energy usage was conducted by utilizing a sample of 50 VMs. The Genetic Algorithm resulted in an average energy consumption of 200.5 kJ for a workload of 500 tasks. However, the WWO-ACO method successfully reduced it to 160.8 kJ.

GA needed 400.2 kJ of energy to do 2500 tasks, while WWO-ACO needed 360.3 kJ. The consistently high performance of WWO-ACO is also clearly shown by other virtual machine configurations. On average, it took the WWO-ACO algorithm 335.0 kJ to run a scenario with 100 virtual machines (VMs) and 2500 tasks. This used much less energy than GA (400.4 kJ) and ACO (344.9 kJ). Furthermore, the GA used 400.2 kJ of energy while handling 2500 tasks, while the WWO-ACO used 360.3 kJ. Also, the excellent performance trend of WWO-ACO is consistently seen in other virtual machine configurations. When examining a sample consisting of 100 virtual machines (VMs) and 2500 tasks, it was found that WWO-ACO had an average energy consumption of 335.0 kJ, which was significantly higher than GA's consumption of 400.4 kJ and ACO's consumption of 344.9 kJ.

Examining the highest point of energy consumption produces equally striking findings. Originally 250.5 kJ for 50 VMs and 500 tasks in GA, WWO-ACO effectively dropped it to 210.8 kJ. The WWO-ACO consumption stays constant while the work grows. For 2500 jobs, WWO-ACO ate 410.3 kJ; GA ate 450.2 kJ. The WWO-ACO algorithm ate a maximum of 410.7 kJ for 200 VMs handling 2500 tasks. This energy consumption was lower than that of the ACO algorithm (425.8 kJ) and the GA algorithm (430.9 kJ). The experimental results reveal that the proposed WWO-ACO method routinely beats conventional optimisation methods regarding average and maximum energy consumption. This shows increased energy efficiency; thus, WWO-ACO has become a feasible method for resource management and workload offloading in cloud computing environments.

Fourth scenario (using resource consumption (RC))

This scenario mainly aims to evaluate the performance of the proposed hybrid WWO-ACO and the existing algorithm's efficiency in terms of RC. An optimum utilisation of resources is always desirable. An RC mainly includes consuming various computing resources such as CPU, memory, and storage. The main goal of the

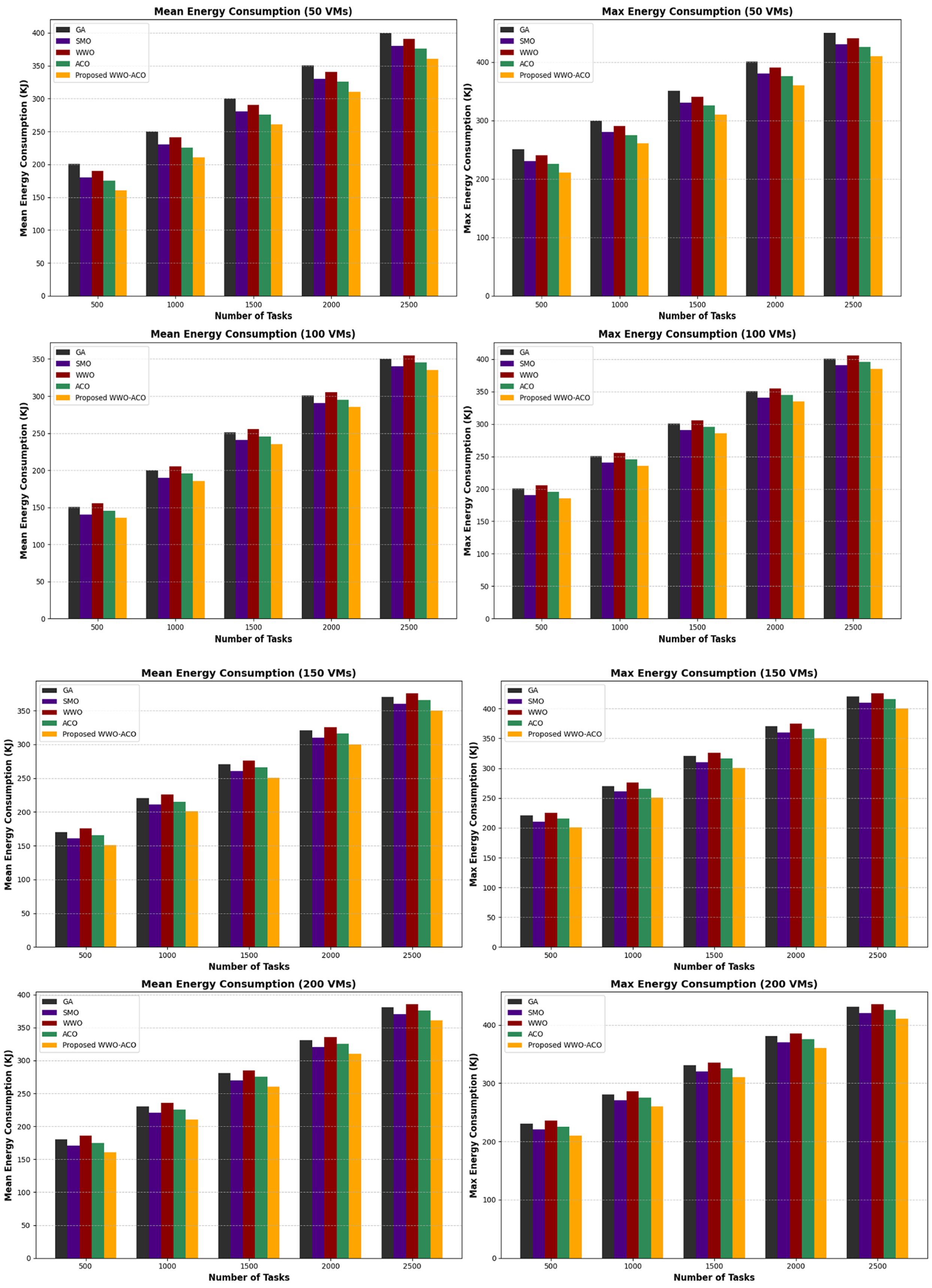

Fig. 3. Comparative analysis of existing and proposed algorithm-based energy consumption (KJ).

proposed method is to reduce resource consumption by maintaining overloading and underloading task execution through efficiently allocating computing resources. This scenario calculates results based on various workload scenarios for existing and proposed algorithms.

The resource utilisation of different optimisation strategies is illustrated in Fig. 4, which also shows how successful the Proposed WWO-ACO Optimization method is in comparison to other traditional methods such as GA and ACO. The proposed optimisation method spreads work and uses fewer resources across configurations. The proposed method can be applied to more virtual machines without losing efficiency because it uses the same resources each time. According to the study, the proposed method reduces resource requirements and maintains performance in various situations.

Fifth scenario (using balance degree)

The hybrid WWO-ACO algorithm is evaluated in Scenario-V using the Balance Degree metric, which measures the evenness of job distribution across resources. A greater Balance Degree guarantees the equitable utilisation of resources, preventing any resource from being overloaded while others are underutilised. This improves the system's stability and effectiveness. The effectiveness of the hybrid algorithm in attaining optimal load distribution, hence enhancing overall system efficiency and dependability, is evaluated by comparing it with GA, SMO, WWO, and ACO using different workloads.

The Balance Degree results in Fig. 5 show the efficiency and effectiveness of several optimisation strategies over several virtual machine configurations and task loads. With values between 0.70 and 0.78, the Proposed WWO-ACO method shows a better balance for 50 VMs than other approaches, including GA and ACO, which have lower values. When compared to other techniques, WWO-ACO demonstrates exceptional performance with 100 VMs. The Balance Degrees range from 0.77 to 0.85, indicating a fairer and even workload distribution. WWO-ACO outperforms GA, SMO, and other methods, achieving values ranging from 0.82 to 0.90 with 150 VMs. The Proposed WWO-ACO method maintains its optimal performance at 200 virtual machines (VMs) with Balance Degrees ranging from 0.87 to 0.95, illustrating its strong capacity to efficiently manage resources and distribute the load evenly across the network. These findings confirm that the Proposed WWO-ACO method consistently achieves a superior balance, rendering it a more practical approach to workload management in various scenarios.

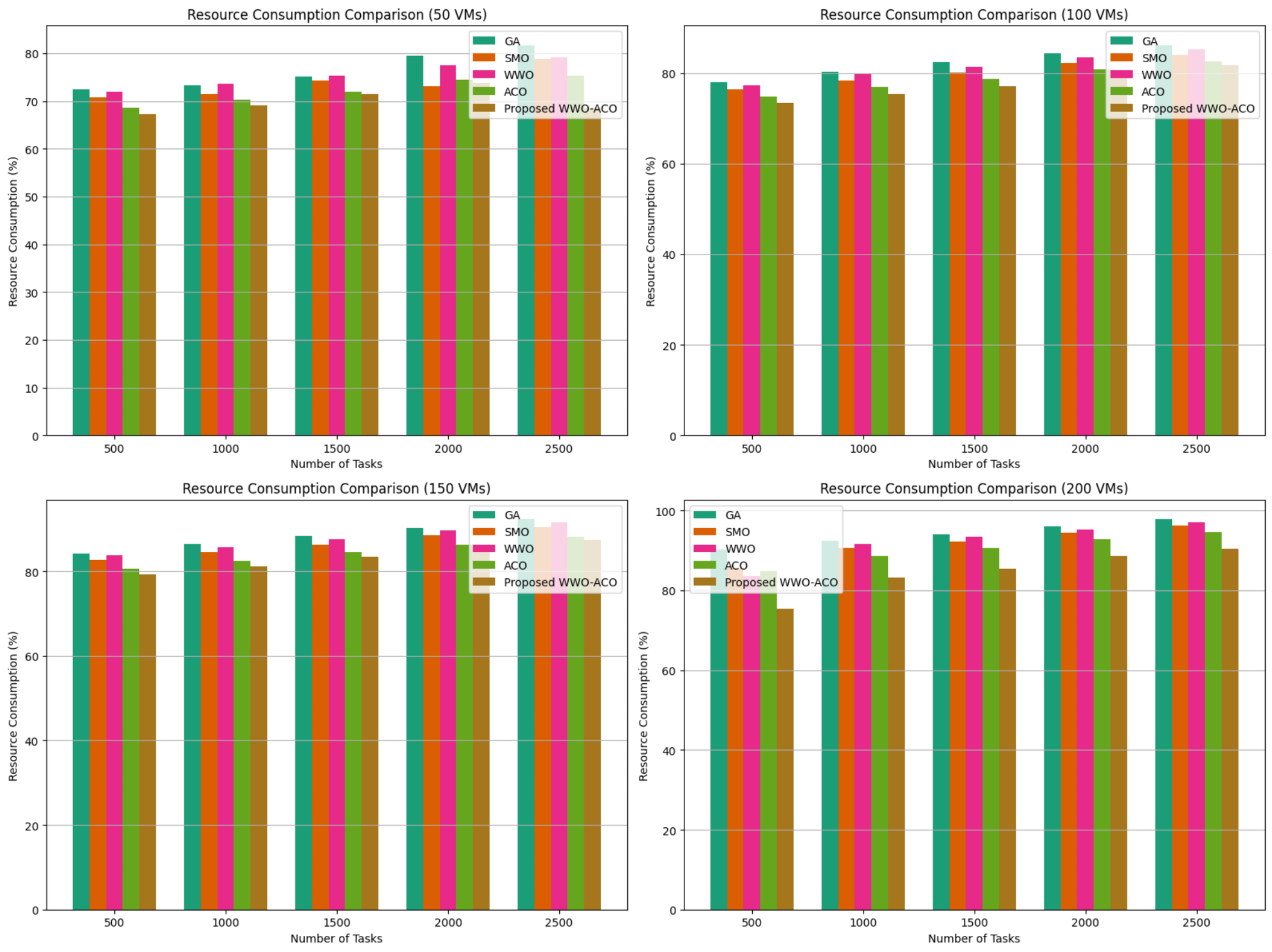

Fig. 4. Comparative analysis of existing and proposed algorithm-based Resource Consumption %

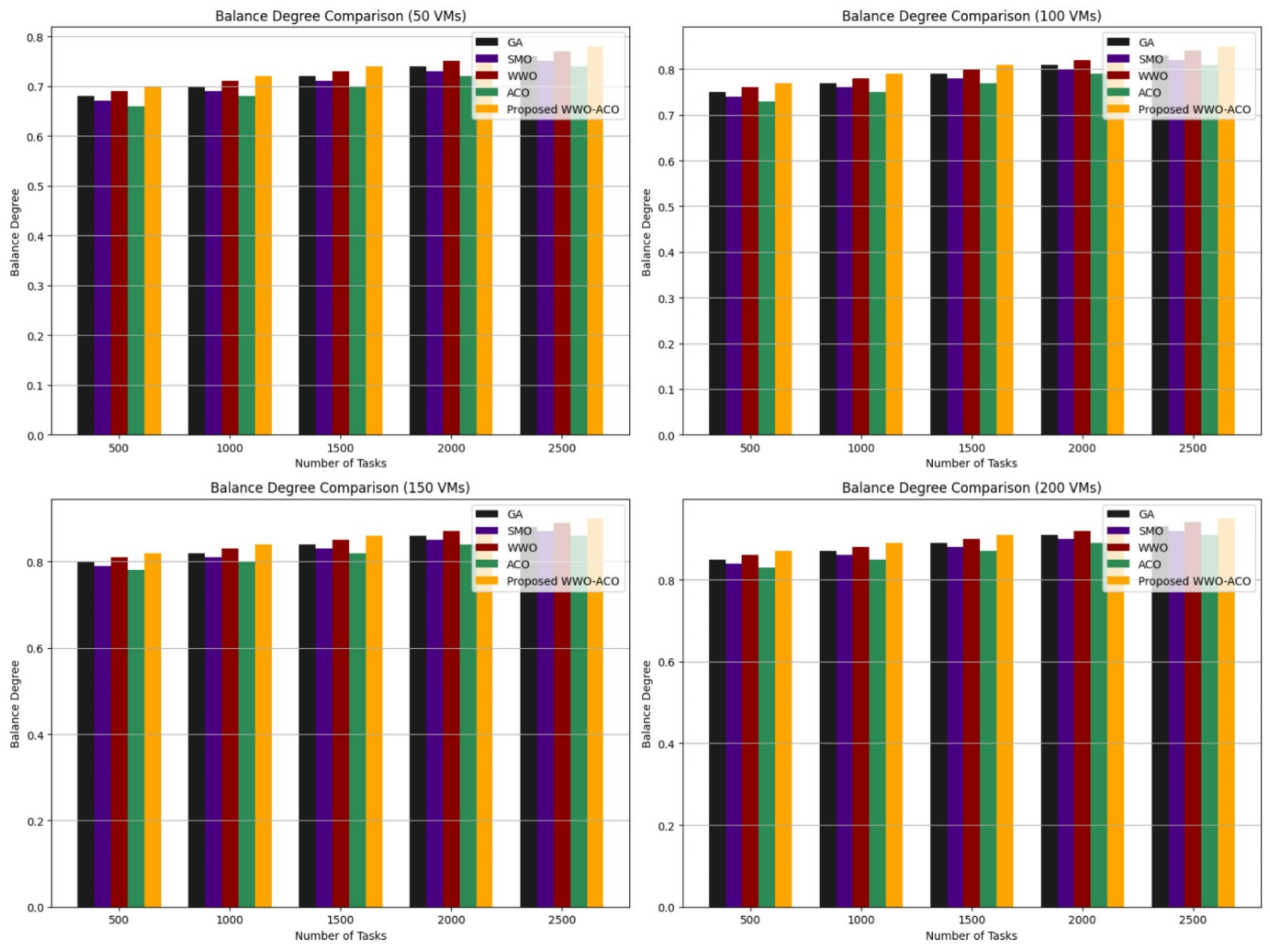

Fig. 5. Comparative analysis of existing and proposed algorithm based on balance degree.

Comparative analysis with state of the art methods

Table 8 provides a comparative analysis of the proposed Hybrid WWO-ACO model against leading methodologies for task scheduling, resource allocation, and load balancing in cloud computing. The table comprises studies from 2023 to 2024, concentrating on diverse optimisation techniques, including the Whale Optimization Algorithm (WOA), Water Wave Optimization (WWO), and their hybrids, such as AGWO (Ant Lion Optimizer combined with WOA) and IACO (Inverted Ant Colony Optimization).

The Hybrid WWO-ACO model surpasses existing methods across various performance metrics, including an 18% reduction in makespan, a 15% decrease in energy consumption, and a 20% enhancement in load balancing. The Hybrid WWO-ACO model, unlike other methods that encounter issues such as restricted scalability, slow convergence, or inadequate adaptability in dynamic environments, integrates the advantages of local search (ACO) and global optimisation (WWO), thereby providing enhanced scalability and adaptability for large-scale, dynamic cloud systems. It more effectively addresses multi-objective optimisation than traditional algorithms, resulting in improved efficiency and overall performance in cloud environments.

Results and discussion

The purpose of this study was to evaluate the effectiveness of a hybrid optimisation method that incorporates WWO and ACO in order to address critical issues that arise in cloud computing environments. These issues include load balancing, task scheduling, and resource allocation. A comparison was carried out between the proposed WWO-ACO hybrid algorithm and several established algorithms, such as the GA, SMO, WWO, and ACO. The evaluation was conducted across five unique scenarios, taking into account task scheduling duration, execution cost, energy consumption, resource utilisation, and load balancing.

- *Task scheduling efficiency*: The preliminary experiment entailed a comparative analysis of task scheduling algorithms, emphasizing both the average and maximum durations of task scheduling. In a situation involving 2000 tasks and 150 virtual machines (VMs), the WWO-ACO hybrid algorithm attained an average task scheduling duration of 1107.8 s (Table 7), indicating a significant enhancement compared to ACO (1407.32 s), SMO (7578.32 s), WWO (8587.7 s), and GA (7378.72 s). The decrease in scheduling time can be linked to the synergistic strengths of ACO's local search abilities and WWO's global exploration efforts. ACO effectively

References	Method	Objective (s)	Key techniques	Optimisation goal	Performance metrics	Challenges addressed	Results
Ghafir et al.[1]	PSO-based Feedback Controller	Load balancing in the cloud	PSO feedback controller	Load distribution optimisation	Load balancing efficiency: 20%, Energy efficiency: 10%	Dynamic load balancing	Improved load balancing in dynamic environments
Dhabliya et al.[2]	Dynamic Load Balancing Policies	Dynamic load balancing in the cloud	Policy-driven strategies	Efficient load balancing	Load balancing improvement: 12%, Response time: 8%	Real-time load balancing	Enhanced load balancing in dynamic environments
Khan[3]	RL-based Clustering for Load Balancing	Dynamic load balancing	RL-based clustering	Optimised task scheduling	Energy reduction: 14%, Load balancing: 10%	Scaling for large systems	Effective dynamic load balancing in cloud systems
Dubey and Mishra[4]	Performance & Trust Analysis for Load Balancing	Load balancing in cloud	Trust and performance evaluation	Trust-based load balancing	Trust evaluation: 16%, Performance improvement: 12%	Trust and security	Improved trust-based load balancing in clouds
Choudhary and Rajak[5]	Min-Min Heuristic for Workflow Scheduling	Workflow scheduling	Min-min heuristic	Task completion time and load balancing	Makespan reduction: 15%, Task completion: 10%	Scalability issues	Efficient for small cloud workflows
Geetha et al.[8]	Hybrid Optimization for Load Balancing	Optimal load balancing	Hybrid optimisation algorithms	Energy and resource optimisation	Energy efficiency: 18%, Load balancing: 14%	Scalability for large systems	Improved load balancing, limited scalability
Forghani et al.[9]	Krill Herd Algorithm for Load Balancing	Load balancing in SDNs	Krill herd metaheuristic	Energy and load balancing	Energy reduction: 20%, Load efficiency: 16%	Network load balancing	Effective in SDNs, limited for the general cloud
Singh et al.[6]	JAYA-based Metaheuristic for Fog-Cloud Ecosystem	Workload distribution in fog-cloud systems	JAYA algorithm for task scheduling	Energy-efficient workload distribution	Energy reduction: 12%, Task completion: 10%	Workload distribution	Effective for fog-cloud systems
Tiwari et al.[7]	Knapsack-based Metaheuristic for Edge Placement	Edge server placement optimisation	Knapsack-based optimisation	Edge server placement	Placement efficiency: 14%, Load balancing: 10%	Edge network optimisation	Effective for edge systems, not scalable to the cloud
Rostami et al.[13]	Capuchin Search & IACO for Task Scheduling	Energy-efficient task scheduling	Capuchin search & IACO	Energy and task scheduling	Energy reduction: 18%, Task completion: 12%	Energy efficiency challenges	Improved task scheduling with energy efficiency
Kumar and Karri[36]	AGWO Hybrid for Task Scheduling	Cost-aware task scheduling	Hybrid Ant Lion & WOA	Cost and task scheduling optimisation	Cost reduction: 14%, Task allocation: 9%	Resource utilisation	Efficient scheduling in cloud-fog systems
Proposed model	Hybrid WWO-ACO	Task scheduling and resource allocation	ACO + WWO	Task completion, load balancing, energy consumption	Makespan reduction: 18%, Energy reduction: 15%, Load balancing: 20%	Scalability and adaptability	Outperforms others in multi-objective optimisation

Table 8. Comparative analysis with State-of-the-art methods.

pinpoints local optima, whereas WWO facilitates a more extensive exploration of the solution space, thereby avoiding suboptimal outcomes. The outcome is a more rapid execution of tasks, enhancing system responsiveness and reducing delays in cloud settings. Additionally, the WWO-ACO hybrid demonstrated a higher load distribution efficiency. The maximum task scheduling time for the WWO-ACO hybrid was 1203.06 s (Table 7), compared to 8301.3 s for GA, 8705.42 s for SMO, 12,150.33 s for WWO, and 1598.72 s for ACO. This demonstrates that the hybrid algorithm enhances overall task scheduling efficiency while optimising load distribution among resources, thereby minimising delays due to resource contention.

- *Execution cost*: The second experiment concentrated on assessing the execution costs linked to each algorithm. The findings indicated that the WWO-ACO hybrid consistently surpassed the other algorithms in reducing task execution expenses. At 150 VMs and 2000 tasks, the WWO-ACO hybrid achieved a mean execution cost of $8301.3 (Fig. 2), which was lower than GA ($7378.72), SMO ($7578.32), WWO ($8587.7), and ACO ($1407.32). The hybrid method's ability to minimise execution costs arises from its efficient balancing of local optimisation (ACO) and global exploration (WWO), which allows resources to be allocated cost-effectively. This minimises excessive expenses while simultaneously guaranteeing the most efficient use of available resources.
- *Energy consumption*: The third experiment also assessed energy consumption. When compared to other algorithms, the WWO-ACO hybrid used significantly less energy. In a configuration of 50 VMs and 500 tasks, the WWO-ACO hybrid used 160.8 kJ (Fig. 3), while the GA used 200.5 kJ. The trend was consistent across configurations, with the WWO-ACO hybrid averaging 335.0 kJ in a configuration of 100 VMs and 2500 tasks, compared to 400.4 kJ for GA. Reducing energy consumption is critical in cloud computing because it reduces costs and promotes sustainability. The WWO-ACO hybrid improves energy efficiency in cloud infrastructure by optimising task scheduling and resource allocation.
- *Resource utilization*: The fourth experiment assessed resource consumption, emphasising the efficiency of each algorithm in utilising available virtual machines. The findings indicated that the WWO-ACO hybrid utilised fewer resources to accomplish the identical set of tasks as the other algorithms. The hybrid approach efficiently distributed the workload among available resources, guaranteeing the completion of necessary tasks without overloading the system. This outcome illustrates that the WWO-ACO hybrid enhances resource utilisation, which is essential for augmenting operational efficiency in cloud settings.
- *Load balancing efficiency*: The fifth experiment assessed the load balancing efficacy of the hybrid WWO-ACO algorithm, employing the Balance Degree metric. The findings demonstrated that the WWO-ACO hybrid consistently attained superior load distribution compared to the other algorithms. With 50 VMs, the Balance Degree for

WWO-ACO ranged from 0.70 to 0.78 (Fig. 5), surpassing that of GA and ACO. The augmentation in the number of VMs enhanced the equilibrium of the WWO-ACO hybrid, achieving a range of 0.87 to 0.95 for 200 VMs. These values indicate that the WWO-ACO hybrid guarantees an equitable allocation of tasks, averting resource saturation and maintaining a stable, efficient system. This optimal load distribution is crucial for enhancing resource efficiency and preventing performance decline.

The results of the experiments demonstrated that the hybrid WWO-ACO algorithm performed better than traditional algorithms in a number of critical areas, such as the scheduling of tasks, the cost of execution, the consumption of energy, the utilisation of resources, and the distribution of load. The success of the hybrid approach can be attributed to the complementary strengths of ACO and WWO. ACO's local search capabilities enable rapid identification of promising solutions, while WWO's global exploration ensures a thorough search of the solution space, avoiding suboptimal configurations. Collectively, these two methodologies offer a robust optimisation instrument for enhancing performance and efficiency in cloud computing settings. This study's findings validate that the hybrid WWO-ACO algorithm is an exceptionally effective method for addressing complex multi-objective optimisation challenges in cloud environments.

Constraints of the proposed hybrid model

There are a few limitations to take into consideration, despite the fact that the hybrid WWO-ACO algorithm has produced some encouraging results:

- *Computational complexity*: The algorithm may require significant computational resources, particularly in extensive or dynamic cloud settings.
- *Performance variability*: The outcome might vary depending on various cloud computing environments and workload variations, necessitating additional adjustments.
- *Adaptability*: The method may find it challenging to accommodate rapidly changing resources and variable workloads.

These constraints offer prospects for future enhancement, specifically in optimizing computational efficiency and improving adaptability in practical applications.

Conclusion and future scope

Multi-objective optimisation plays a vital role in the ever-changing landscape of cloud computing. Conventional single-objective optimisation methods frequently prove inadequate when dealing with the intricacies of contemporary cloud systems. In order to address these limitations, this research presents a cutting-edge multi-objective hybrid optimisation technique that skillfully combines the advantages of ACO and WWO. The hybrid WWO-ACO algorithm has proven exceptionally effective in optimising resource allocation and cloud load balancing. Our empirical evaluations show that the algorithm achieves an average task schedule length of just 1107.8 s with 150 VMs and 2000 tasks. This performance significantly outstrips traditional methods such as GA (7378.72 s), SMO (7578.32 s), WWO (8587.7 s), and ACO (1407.32 s). This reduction in task completion time highlights the hybrid's superior efficiency. Regarding energy consumption, the WWO-ACO algorithm also excels, using an average of only 335.0 kJ for 2500 tasks and 100 VMs, compared to GA's 400.4 kJ and ACO's 344.9 kJ.

In addition, the method demonstrates outstanding resource balancing, as indicated by balance degrees ranging from 0.87 to 0.95 across various configurations. This suggests that the allocation of resources is fair and effective. The study's findings demonstrate the algorithm's ability to enhance system responsiveness in dynamic cloud environments, reduce operational costs, and enhance performance criteria. While the hybrid WWO-ACO model demonstrates impressive performance, it is not without challenges. The complexity of the task may result in increased computational requirements, especially in large or dynamic environments. Moreover, the system's performance may differ depending on the specific cloud infrastructures and workloads, necessitating additional adjustments and optimisations. Future research should address these challenges by reducing the computational load of the model, testing its compatibility with diverse cloud configurations, and developing adaptive mechanisms for fluctuating resource needs. Exploring various optimisation methods and considering real-world constraints will be essential for improving the model's practical utility and efficiency. In the future research, we will also examine how the emerging algorithms, i.e., Algorithms like the Liver Cancer Algorithm (LCA), Fata Morgana Algorithm (FATA), Polar Lights Optimization (PLO), and Rime Optimization Algorithm (RIME) stack up against our proposed hybrid approach, which combines Ant Colony Optimization (ACO) and Whale Optimization Algorithm (WOA).

Data availability

The datasets used and/or analyzed during the current study available from the corresponding author on reasonable request.

Received: 18 October 2024; Accepted: 27 March 2025
Published online: 08 April 2025

References

1. Ghafir, S., Alam, M. A., Siddiqui, F. & Naaz, S. Load balancing in cloud computing via intelligent PSO-based feedback controller. *Sustain. Comput.: Inform. Syst.* **41**, 100948 (2024).
2. Dhabliya, D., Dari, S. S., Sakhare, N. N., Dhablia, A. K., Pandey, D., Muniandi, B., George, A. S., Hameed, A.S. & Dadheech, P. New proposed policies and strategies for dynamic load balancing in cloud computing. In *Emerging trends in cloud computing analytics, scalability, and service models* pp. 135–143 (IGI Global, 2024).

3. Khan, A. R. Dynamic load balancing in cloud computing: Optimized RL-based clustering with multi-objective optimized task scheduling. *Processes* **12**(3), 519 (2024).
4. Dubey, A. K. & Mishra, V. Analysis of performance and trust in load balancing algorithm in cloud computing environment. *Int. J. Adv. Intell. Paradigms* **27**(2), 91–103 (2024).
5. Choudhary, A. & Rajak, R. A novel strategy for deterministic workflow scheduling with load balancing using modified min-min heuristic in cloud computing environment. *Cluster Comput.*, pp. 1–22 (2024).
6. Singh, S., Sham, E. E. & Vidyarthi, D. P. Optimizing workload distribution in Fog-Cloud ecosystem: A JAYA based meta-heuristic for energy-efficient applications. *Appl. Soft Comput.* **154**, 111391 (2024).
7. Tiwari, V., Pandey, C., Dahal, A., Roy, D. S. & Fiore, U. A knapsack-based metaheuristic for edge server placement in 5G networks with heterogeneous edge capacities. *Futur. Gener. Comput. Syst.* **153**, 222–233 (2024).
8. Geetha, P., Vivekanandan, S. J., Yogitha, R. & Jeyalakshmi, M. S. Optimal load balancing in cloud: Introduction to hybrid optimization algorithm. *Expert Syst. Appl.* **237**, 121450 (2024).
9. Forghani, M., Soltanaghaei, M. & Boroujeni, F. Z. Dynamic optimization scheme for load balancing and energy efficiency in software-defined networks utilizing the krill herd meta-heuristic algorithm. *Comput. Electr. Eng.* **114**, 109057 (2024).
10. Li, Z. et al. Tactical unit algorithm: A novel metaheuristic algorithm for optimal loading distribution of chillers in energy optimization. *Appl. Therm. Eng.* **238**, 122037 (2024).
11. Singhal, S., Sharma, A., Verma, P. K., Kumar, M., Verma, S., Kaur, M., Rodrigues, J. J., Khurma, R. A. & García-Arenas, M. Energy efficient load balancing algorithm for cloud computing using rock hyrax optimization. *IEEE Access* (2024).
12. Priyadarshi, R. Energy-efficient routing in wireless sensor networks: a meta-heuristic and artificial intelligence-based approach: A comprehensive review. *Arch. Computat. Methods Eng.* **31**(4), 2109–2137 (2024).
13. Rostami, S., Broumandnia, A. & Khademzadeh, A. An energy-efficient task scheduling method for heterogeneous cloud computing systems using capuchin search and inverted ant colony optimization algorithm. *J. Supercomput.* **80**(6), 7812–7848 (2024).
14. Hou, H., Jawaddi, S. N. A. & Ismail, A. Energy efficient task scheduling based on deep reinforcement learning in cloud environment: A specialized review. *Futur. Gener. Comput. Syst.* **151**, 214–231 (2024).
15. Khaleel, M. I. Region-aware dynamic job scheduling and resource efficiency for load balancing based on adaptive chaotic sparrow search optimization and coalitional game in cloud computing environments. *J. Netw. Comput. Appl.* **221**, 103788 (2024).
16. Simaiya, S. et al. A hybrid cloud load balancing and host utilisation prediction method using deep learning and optimisation techniques. *Sci. Rep.* **14**(1), 1337 (2024).
17. Kak, S. M., Agarwal, P., Alam, M. A. & Siddiqui, F. A hybridized approach for minimizing energy in cloud computing. *Clust. Comput.* **27**(1), 53–70 (2024).
18. Behera, I. & Sobhanayak, S. Task scheduling optimization in heterogeneous cloud computing environments: A hybrid GA-GWO approach. *J. Parallel Distributed Comput.* **183**, 104766 (2024).
19. Verma, G. Hybrid optimization model for secure task scheduling in cloud: Combining seagull and black widow optimization. *Cybern. Syst.* **55**(8), 2489–2511 (2024).
20. Ahmed, A., Adnan, M., Abdullah, S., Ahmad, I., Alturki, N. & Menzli, L. J. An efficient task scheduling for cloud computing platforms using energy management algorithm: A comparative analysis of workflow execution time. *IEEE Access* (2024).
21. Shrivastava, P., Alam, B. & Alam, M. A hybrid lightweight blockchain based encryption scheme for security enhancement in cloud computing. *Multimedia Tools Appl.* **83**(1), 2683–2702 (2024).
22. Narayanan, G. & Kannan, S. Enhancing security in cloud-based VM migration: A trust-centric hybrid optimization approach. *Int. Arab J. Inf. Technol.* **21**(1), 166–177 (2024).
23. Hongwei, D. I. N. G. & Zhang, Y. Efficient cloud workflow scheduling with inverted ant colony optimization algorithm. *Int. J. Adv. Comput. Sci. Appl.* **14**(10) (2023).
24. Sugan, J. PredictOptiCloud: A hybrid framework for predictive optimisation in hybrid workload cloud task scheduling. *Simul. Model. Pract. Theory* **134**, 102946 (2024).
25. Khan, Z. A. & Izzatdin, A. A. Ripple-induced whale optimisation algorithm for independent tasks scheduling on fog computing. *IEEE Access* (2024).
26. Geeta, K. & Kamakshi Prasad, V. Multi-objective cloud load-balancing with hybrid optimisation. *Int. J. Comput. Appl.* **45**(10), 611–625 (2023).
27. Medara, R. & Singh, R. S. Dynamic virtual machine consolidation in a cloud data center using modified water wave optimisation. *Wireless Personal Commun.* **130**(2), 1005–1023 (2023).
28. Sumathi, M., Vijayaraj, N., Raja, S. P. & Rajkamal, M. HHO-ACO hybridised load balancing technique in cloud computing. *Int. J. Inform. Technol.* **15**(3), 1357–1365 (2023).
29. Malathi, K. & Priyadarsini, K. Hybrid lion–GA optimisation algorithm-based task scheduling approach in cloud computing. *Appl. Nanosci.* **13**(3), 2601–2610 (2023).
30. Janakiraman, S. & Deva Priya, M. Hybrid grey wolf and improved particle swarm optimisation with adaptive intertial weight-based multi-dimensional learning strategy for load balancing in cloud environments. *Sustain. Comput.: Inform. Syst.* **38**, 100875 (2023).
31. Kumar, K. V. & Rajesh, A. Multi-objective load balancing in cloud computing: A meta-heuristic approach. *Cybern. Syst.* **54**(8), 1466–1493 (2023).
32. Neelakantan, P. & Sudhakar Yadav, N. An optimised load balancing strategy for an enhancement of cloud computing environment. *Wireless Personal Commun.* **131**(3), 1745–1765 (2023).
33. Kaur, A. & Kaur, B. Load balancing optimisation based on hybrid Heuristic-Metaheuristic techniques in cloud environment. *J. King Saud Univer.–Comput. Inform. Sci.* **34**(3), 813–824 (2022).
34. Zhou, J. et al. Comparative analysis of metaheuristic load balancing algorithms for efficient load balancing in cloud computing. *J. Cloud Comput.* **12**(1), 85 (2023).
35. Chen, M.-R., Zeng, G.-Q. & Lu, K.-D. A many-objective population extremal optimization algorithm with an adaptive hybrid mutation operation. *Inf. Sci.* **495**, 331–350. https://doi.org/10.1016/j.ins.2019.02.053 (2019).
36. Kumar, M. S. & Karri, G. R. AGWO: Cost aware task scheduling in cloud fog environment using hybrid metaheuristic algorithm. *Int. J. Exp. Res. Rev* **33**, 41–56 (2023).

Acknowledgements
The authors extend their appreciation to King Saud University for funding this work through Researchers Supporting Project number (RSPD2025R685), King Saud University, Riyadh, Saudi Arabia. The authors extend their appreciation to the Deanship of Scientific Research at King Khalid University for funding this work through large group Research Project under grant number. RGP 2/218/45.The author would like to thank the Deanships of Scientific Research at Shaqra University for supporting this work.

Author contributions
Umesh Kumar Lilhore led the conceptualization and methodology, ensuring the research's overall integrity. Yogendra Narayan Prajapati conducted the data analysis and interpretation. Anjani Kumar Rai and Sarita Simaiya

contributed to writing and revising the manuscript, while Ehab Seif Ghith, Mehdi Tlija, and Tarik Lamoudan provided critical insights and feedback. Abdelaziz A. Abdelhamid supported the project through resource management and coordination.

Declarations

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to S.S. or A.A.A.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by-nc-nd/4.0/.

© The Author(s) 2025