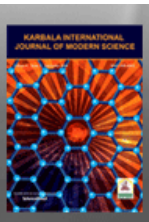

Karbala International Journal of Modern Science

Manuscript 3396

An Effective Secure Multi-Objective Task Scheduling Algorithm in Multi-Cloud Environment

V K S K Sai Vadapalli

Ramesh Babu Gurujukota

Phaneendra Varma Chintalapati

Satyanarayana Murty

G. Sai Chaitanya Kumar

See next page for additional authors

An Effective Secure Multi-Objective Task Scheduling Algorithm in Multi-Cloud Environment

Abstract

In cloud environments, task scheduling is essential for improving performance. Nevertheless, the existence of several heterogeneous clouds makes scheduling extremely difficult, requiring increasingly advanced algorithms to manage these environments' diversity and dynamic nature. To solve this, numerous authors have created a variety of task schedulers utilizing heuristic and metaheuristic techniques. Nevertheless, it remains dynamic and challenging because task scheduling is an NP-hard issue. Furthermore, in many complicated situations, it is still problematic to guarantee security throughout the task's execution. Therefore, this paper introduces a multi-objective security-aware task scheduler using the Crayfish Mud Ring Optimization Algorithm for a multi-cloud environment. This approach combines the Mud Ring Optimization approach and the Crayfish Optimization Algorithm to improve the results and overcome their shortcomings. Due to this hybridization, the proposed algorithm can avoid local optima and converge toward globally optimal solutions. Thus, the suggested scheduling algorithm offers the best work allocation, considering four objectives: makespan, cost energy, and security limitations (risk assessment). Comprehensive simulations are run on two real-world workloads, such as High-Performance Computing Center North and NASA Ames iPSC/860, and outperforming all comparison algorithms with an average improvement of 46% makespan, 55% energy, 61% cost, and 52% security risk in all the examined scenarios. The outcomes show that the suggested CMROA performs better and is suitable for a multi-cloud environment.

Keywords

CMROA, task scheduling, multi-cloud, NASA, HPC2N

Authors

V K S K Sai Vadapalli, Ramesh Babu Gurujukota, Phaneendra Varma Chintalapati, Satyanarayana Murty, G. Sai Chaitanya Kumar, and Satish Kumar Kode

RESEARCH PAPER

An Effective Secure Multi-objective Task Scheduling Algorithm in Multi-cloud Environment

V.K.S.K. Sai Vadapalli [a,*], Ramesh babu Gurujukota [b], Phaneendra varma chintalapati [b], P.T. Satyanarayana Murty [b], G. Sai Chaitanya Kumar [c], Satish kumar kode [b]

[a] Department of IT, SRKR Engineering College, Bhimavaram, West Godavari District, Andhra Pradesh, India
[b] Department of Computer Science and Engineering, Shri Vishnu Engineering College for Women (A), Bhimavaram, West Godavari District, Andhra Pradesh, India
[c] Department of Artificial Intelligence, DVR&Dr. HS MIC College of Technology, Kanchikacherla, Andhra Pradesh, India

Abstract

In cloud environments, task scheduling is essential for improving performance. Nevertheless, the existence of several heterogeneous clouds makes scheduling extremely difficult, requiring increasingly advanced algorithms to manage these environments' diversity and dynamic nature. To solve this, numerous authors have created a variety of task schedulers utilizing heuristic and metaheuristic techniques. Nevertheless, it remains dynamic and challenging because task scheduling is an NP-hard issue. Furthermore, in many complicated situations, it is still problematic to guarantee security throughout the task's execution. Therefore, this paper introduces a multi-objective security-aware task scheduler using the Crayfish Mud Ring Optimization Algorithm for a multi-cloud environment. This approach combines the Mud Ring Optimization approach and the Crayfish Optimization Algorithm to improve the results and overcome their shortcomings. Due to this hybridization, the proposed algorithm can avoid local optima and converge toward globally optimal solutions. Thus, the suggested scheduling algorithm offers the best work allocation, considering four objectives: makespan, cost energy, and security limitations (risk assessment). Comprehensive simulations are run on two real-world workloads, such as High-Performance Computing Center North and NASA Ames iPSC/860, and outperforming all comparison algorithms with an average improvement of 46 % makespan, 55 % energy, 61 % cost, and 52 % security risk in all the examined scenarios. The outcomes show that the suggested CMROA performs better and is suitable for a multi-cloud environment.

Keywords: CMROA, HPC2N, Multi-cloud, NASA, Task scheduling

1. Introduction

With cloud computing, programs and processing power can be made available over the Internet as services whenever needed. It has garnered significant attention from both academics and corporations. It is made possible by virtual machine technologies' adaptability and affordable processing power [1−3]. The rapid expansion and advancement of Internet of Things (IoT) technology has resulted in a tremendous increase in data volume, raising the need for computing resources. However, due to their inherent restrictions, traditional single-cloud setups are inadequate in meeting user expectations. Due to these reasons, Multi-Cloud Environments (MCEs) have attracted much interest from academics and industry alike [4−6]. They provide various virtual resources, enhance service quality, avoid vendor lock-in and single points of failure, and more. As a result, consumers have many options, and their need for Quality of Service (QoS) is better met.

However, managing and optimizing jobs across various clouds can be challenging, as it involves determining the ideal times and places for executing tasks to achieve performance goals. Cost,

Received 29 July 2024; revised 6 January 2025; accepted 9 January 2025.
Available online 28 January 2025

* Corresponding author.
E-mail address: vksksaivadapalli@gmail.com (V.K.S.K.S. Vadapalli).

https://doi.org/10.33640/2405-609X.3396

performance, data location, and security across various cloud providers are some variables that make scheduling in MCE more complex [7–9]. As a result, the main objective is to allocate work to the best cloud resources according to resource availability, workload features, and user-defined policies. It entails distributing the burden evenly, making the most use of available resources, cutting expenses and energy, and enhancing performance across many clouds. Therefore, choosing a proper task scheduler to schedule the task is a primary challenge [10–12]. It assists users in carrying out tasks using appropriate virtual resources, guaranteeing SLA adherence and excellent service. An efficient task scheduler benefits cloud service providers and customers by improving cloud-based services' general effectiveness and dependability.

Previously, various methods have been developed to address the issue of job scheduling. Initially, several heuristics-based methods were created to help with job scheduling issues. Nevertheless, when the problem size and the number of goals to be optimized grow, the accuracy of solutions produced by these heuristic procedures drastically declines. Furthermore, it is heavily dependent upon a set of underlying rules, the generation of which comes at a significant running cost [13–15]. On the other hand, metaheuristic methods have shown themselves to be reliable and efficient in resolving a broad range of challenging optimization issues in the actual world. It can be explained by the fact that, unlike heuristic techniques, they can use a set of candidate solutions to traverse the solution space instead of only one. While some of these algorithms have demonstrated encouraging progress in identifying the globally best solution for the cloud-based task scheduling problem, most suffer from premature convergence and have trouble overcoming local minima, mainly when the solution space is vast [16–18]. These restrictions frequently result in less-than-ideal job scheduling strategies, which impair system performance and go against QoS assurances.

Furthermore, conventional task scheduling algorithms mainly focused on makespan and ignored the cost and energy consumption. Therefore, this paper's main objective is to consider makespan, energy usage, and cost. Security in the cloud is another crucial factor that must be considered [19,20]. Because the cloud offers a shared infrastructure for many users, data security must be established while scheduling tasks to prevent unwanted access to essential data files, and a lack of data security could result in data tampering or delay. Most of the existing task scheduling algorithms offer a multi-objective scheduling strategy that minimizes only execution time and does not consider security as one of the QoS dimensions. They are not suitable for use in a risky distributed system.

Therefore, in this work, an effective hybrid Crayfish Mud Ring Optimization Algorithm (CMROA) is implemented to perform security-aware job scheduling in a multi-cloud environment. In this hybrid algorithm, we combine the exploration phase of the crayfish optimization algorithm (COA) and the exploitation phase of the mud ring optimization algorithm (MROA) to overcome the poor convergence rate and easily trap into local optima. Combining these two stages, CMROA guarantees a balance between exploitation (adjusting task assignments for maximum security and performance) and exploration (looking for varied job allocations), which is often challenging when using single methods. The primary goal of this hybrid CMROA technique is to effectively balance energy consumption, costs, time, and security while scheduling tasks in the multi-cloud system. Due to its robustness, CMROA is better suited for settings with varying workloads and resources. Due to the scalable nature of both COA and MROA, CMROA's modular design enables it to manage more tasks without declining efficiency. Multiple tasks are processed concurrently in each step, enabling CMROA to sustain performance levels even as the number of tasks rises.

The main contributions of this paper can be summarized as follows.

- This work suggests a novel hybrid task scheduling algorithm, the Crayfish Mud Ring Optimization Algorithm, by combining the exploration phase of the Crayfish Optimization Algorithm with the exploitation phase of the Mud Ring Optimization approach, which successfully balances exploration and exploitation for better task allocation in multi-cloud circumstances.
- The proposed algorithm introduces a security-aware scheduling framework, addressing multi-cloud environments' security requirements while minimizing the execution cost, energy, and makespan.
- Using real-world workloads like NASA and HPC2N, the suggested strategy's effectiveness is evaluated compared to alternative scheduling strategies.

The remaining portions are arranged as follows: the analysis of relevant works is described in Section 2, and Section 3 presents the preliminaries. In

Section 4, we formulate the task scheduling problem; in Section 5, we present the proposed hybrid algorithm to solve the scheduling problem. Several simulated experiments and the results are given in Section 6, and Implications, Practical Benefits and real-world applications are given in Section 7. Potential Integration challenges are given in section 8, and limitations and future directions are given in section 9. Finally, the conclusion is presented in section 10.

2. Literature review

This section discusses the various parameters and development methodologies used to create schedulers for the cloud computing and multi-cloud computing paradigms.

To schedule the tasks in cloud computing, Abu-Hashem et al. [21] introduced the black widow optimization algorithm. One of the significant drawbacks of the black widow algorithm was the random selection process. Therefore, the author introduced several selection methods to solve this problem, such as Survivor, Truncation, Tournament, Exponential ranking, linear ranking, stochastic universal selection, etc. It is important to note that every selection technique has advantages and disadvantages. Thus, each version's efficiency was assessed using the CEC 2019 benchmark dataset, and the results were compared with existing methods. Furthermore, cloud-specific parameters like makespan, energy, and cost were used in the comparisons.

In another work, Mangalampalli et al. [22] presented an asynchronous advantage actor-critic (a3c) algorithm based on a multi-objective prioritized task scheduler in a multi-cloud environment. In this work, two steps were involved in the scheduling procedure. Task managers determine the VM priorities for all incoming tasks in the initial phase. Then, in the second stage, the priorities were fed into the suggested scheduler, which creates decisions about scheduling to assign tasks to virtual machines (VMs). During this process, the scheduler considers the priorities and schedules tasks based on costs, resource utilization, and makespan in the available multi-cloud environment. The authors used NASA and HPC2N datasets for performance assessment, and the results were compared with existing techniques.

To address the scheduling issues, three heuristic algorithms such as PTMin-Max, PTMax-Min, and Pair-Task Threshold Limit (PTL) were suggested by Krishnasamy et al. [23]. The suggested heuristics first compute the task threshold value based on the task attributes to establish the task scheduling order. Following this, the tasks were separated into two groups. In the first group, the tasks are arranged according to decreasing threshold values, and the second group contains the remaining tasks ([n/2] − 1) arranged according to the threshold's ascending value. After that, depending on the minimum completion time, tasks from Group 1 are scheduled first, followed by tasks from Group 2.

In another work, the multi-task, multi-objective optimization approach was presented by Yi et al. [24] to optimize scheduling in a multi-cloud environment. In this work, the tasks were divided into CPU-intensive jobs, which require a lot of calculation and CPU resources, and I/O-intensive tasks, which frequently require memory access, based on the various data characteristics of tasks in MCE. Next, to schedule these two activities concurrently, the authors implemented a multi-objective and multi-factor evolutionary algorithm based on quadratic crossover (I-MOMFEA-II). Finally, the CEC17-MTMO dataset assessed this method's scheduling efficiency regarding throughput, cost and energy.

Huang et al. [25] created a hybrid meta-heuristic algorithm by integrating the genetic algorithm (GA) and particle swarm optimization (PSO) for security-aware task scheduling in hybrid clouds. To get over GA's slow rate of convergence and PSO's tendency to become easily stuck in local optima, this hybrid algorithm uses the self and social cognition for the evolution framework of PSO and the evolution operators of GA to reach the optimal assignment. To be more precise, this algorithm represents a task–resource assignment using the code (each chromosome's gene value, particle location) of each individual (GA's chromosomes, PSO' particle). It redistributes jobs with incomplete requirements from one resource to another to enhance the performance of decoded assignments. It increases the quantity of accepted tasks, enhancing user satisfaction.

To guarantee work scheduling and load balancing in cloud environments, Elsakaan and Amroun [26] introduced a hybrid method. It contains three primary phases. In the first phase, servers with comparable occupation characteristics were divided into clusters with a bounded size using a k-means-based algorithm. Then, the round-robin algorithm was utilized to select which cluster was executed first. After that, the genetic algorithm was introduced to choose which task is executed first in the cluster. Furthermore, in the last phase, the algorithm determines which machines will be released from which cluster. The cloudlets were then obtained and

routed to the global tasks scheduler module for additional planning. The authors of the algorithm tested their algorithms under many conditions, including cluster sizes, cloudlet, and server, to ensure the accuracy of the results.

By incorporating the security-prioritized mapping scheme for cloud computing environments, Alam et al. [27] presented a security-prioritized multiple workflow allocation (SPMWA) model. To reduce the likelihood of a cloud system failure, this approach prioritizes jobs requiring high levels of security while allocating resources to more reliable virtual machines. One way to reduce the likelihood of failure is to assign tasks to reliable virtual machines with a high enough trust level to reduce the number of task failures. To compare the SPMWA, the number of task failures, failure probability, and makespan have all been calculated.

Gu et al. [28] presented a deep learning-based technique to enhance resource efficiency and workload offloading in a mobile edge computing setting. Deep neural networks and genetic algorithms were used in this work to maximize offloaded jobs while minimizing latency and energy consumption. It initially stores comprehensive information about each case and employs a Genetic Algorithm to determine the best offloading options. A neural network is then trained using these data to make efficient offloading decisions without rerunning GA.

A Chameleon and Remora Search Optimization Algorithm was proposed by Pabitha et al. [29] to achieve an effective scheduling process by investigating the effects of network bandwidth and MIPS. Additionally, throughout the scheduling process, the job simultaneously incorporates variables such as makespan, scheduling cost, load balance, and task completion rate. The advantages of the Chameleon Search Algorithm and Remora Search Optimization Algorithm were combined with the greedy approach to create a multi-objective cloud task scheduling optimization model.

To create schedules, Mangalampalli et al. [30] developed an adaptive task scheduler that divides all jobs that arrive at the cloud interface into smaller tasks and feeds them to the scheduler, which is modelled by the Improved Asynchronous Advantage Actor-Critic Algorithm (ATSIA3C). There were two steps involved in this scheduling procedure. In the initial phase, every task that came in was divided into smaller assignments. Following segmentation, all of these smaller jobs were grouped and fed into the ATSIA3C scheduler based on factors including size, completion time, and interaction time. In the second step, it looks for the aforementioned constraints and distributes them across the datacenter-based VMs with the appropriate processing capability.

In another work, an Efficient Multiverse Electro Search optimization was developed by Ravi and Pillai [31] to enhance work scheduling behaviour. This optimization worked based on cosmological concepts and electron transport close to the nucleus. To find the best option from a vast number of viable options, this approach combines local search, exploitation, and exploration. It utilized the idea of a multiverse to explore a wide search area and find the best solutions for job scheduling and resource allocation. Finally, the authors evaluated the effectiveness of the suggested technique using Inspiral, SIPHT, Montage, and CyberShake workloads and compared it with existing methods.

Lwin [32] used a multi-stage optimization strategy to increase scheduling and allocation performance. There are three phases in this work. During the initial phase, the system determined four critical metrics: Total Task Length, Earliest Finished Time of a specific VM, Communication Cost, and Computation Cost. During the second phase, the K-Means clustering technique was used to group tasks. In the last phase, the authors developed a resource allocation method that assigns the grouped jobs to the relevant virtual machines. By assigning the tasks to the most appropriate resources, this stage optimizes system efficiency and resource usage.

Table 1 summarises the literature review and the drawbacks of existing methods. By combining the exploration and exploitation stages of MROA and COA, our suggested CMROA approach overcomes these difficulties. By simulating various search tactics, COA's exploration improves global search capabilities and encourages deeper solution space investigation. The exploitation phase of MROA, on the other hand, effectively manages multi-objective optimization by classifying solutions according to Pareto dominance, which enables our approach to find the best trade-offs between competing goals. Compared with existing techniques, this hybrid CMROA methodology guarantees faster and more reliable convergence. Additionally, our approach is flexible and scalable, making it ideal for complicated, high-dimensional optimization tasks and big datasets. Additionally, it provides an effective way to get around the drawbacks of conventional optimization techniques, resulting in better performance.

3. Preliminary

This section covers the basic principles of the COA and MROA algorithms. In the proposed study,

Table 1. Summary of literature review.

Paper	Technique	Tool used to simulate	Environment	Dataset	Makespan	Energy	Cost	Security risk	Limitations
Hashem et al. [21]	Black widow optimization algorithm	MATLAB	Cloud	CEC 2019 benchmark dataset	✓	×	✓	×	It is limited to local optima because of its early convergence.
Mangalampalli et al. [22]	MOPTSA3C	Cloudsim	Multi-cloud	HPC2N, NASA	✓	×	✓	×	This work proposed a two-stage scheduling strategy, which resulted in significant overhead. Additionally, the algorithm requires a lot of computing power.
Krishnasamy et al. [23]	PTMin-Max, PTMax-Min, and Pair-Task Threshold Limit (PTL)	Not mentioned	Multi-cloud	12 different types of task	✓	×	×	×	The use of numerous methods, including PTL, PTMin-Max, and PTMax-Min, results in increased resource use and computational overhead.
Yi et al. [24]	MTMO	Not mentioned	Multi-cloud	CEC17 dataset	×	✓	✓	×	Before scheduling, classifying jobs as CPU or I/O-based may cause delays and increased overhead.
Huang et al. [25]	Hybrid GA and PSO	MATLAB	Hybrid cloud	Randomly generated task	×	✓	✓	×	Evaluation takes only a small number of tasks.
Elsakaan et al. [26]	k-means and round-robin	Cloudsim	Cloud	Group of tasks	✓	×	×	×	Makepan is the only main criterion taken for evaluation
Alam et al. [27]	SPMWA	MATLAB	Cloud	Montage, CyberShake, and LIGO	✓	×	×	✓	For task execution, it only prioritizes high-security virtual machines, which could result in increased running costs.
Gu et al. [28]	DNN-GA	Not mentioned	Mobile edge computing	Own dataset	×	✓	✓	×	Increased computational cost
Pabitha et al. [29]	Chameleon and Remora Search Optimization Algorithm	CloudSim	Cloud	Group of tasks	✓	✓	✓	×	A greedy approach utilized in this work might result in inefficient outcomes because it can become trapped in local optima, particularly in a dynamic environment.
Mangalampalli [30]	ATSIA3C	CloudSim	Multi-cloud	HPC2N, NASA	✓	✓	✓	×	Task division into smaller components may result in more complexity, particularly for tasks with complex relationships, which may cause performance inefficiency.
Ravi and Pillai [31]	Multiverse Electro Search optimization	Python-based simulation environment	Multi-cloud	Inspiral, SIPHT, Montage, and CyberShake	✓	×	✓	×	The approach is computationally demanding due to the usage of intricate ideas such as the multiverse and electrostatic forces.
Lwin [32]	K-Means clustering	CloudSim	Cloud	Randomly generated task	✓	×	×	×	K-Means clustering may not be appropriate for dynamic workloads since it demands a predetermined number of clusters.
Proposed	CMROA	CloudSim	Multi-cloud	HPC2N, NASA	✓	✓	✓	✓	–

a combination of these algorithms provides the backbone for handling intricate scheduling tasks in a multi-cloud environment.

3.1. Overview of crayfish optimization algorithm

The COA is a novel metaheuristic algorithm that draws inspiration from the natural foraging habits of crayfish. Fundamentally, COA is organized around three different behavioural phases that each replicate a different facet of the crayfish life cycle: the competition stage for resource distribution, the foraging stage for exploitation, and the summer resort period for exploration. The summer resort stage directs the algorithm to investigate various areas of the solution space to avoid premature convergence, symbolizing the search for a more astonishing residence. The algorithm's capacity to amplify the search in attractive locations is demonstrated by the competition stage when crayfish compete for resources. Lastly, the foraging stage—comparable to crayfish searching for food—represents the algorithm's solution development. The COA is unique because its temperature-based mechanism regulates the changes between these phases.

3.2. Overview of mud ring optimization algorithm

Tursiops truncates, or bottlenose dolphins, are one of 76 marine mammals called cetaceans. Their diving depth is around 260 m into the ocean's surface. Bottlenose dolphins collaborate by maximizing their hunting effort as a team to obtain food. Dolphins employed a variety of hunting techniques. The prey items and ambient factors (habitat) influence these tactics. Mud Ring feeding, or mud plume shing, is a unique foraging. In this method of foraging, a single dolphin from the crowd swims around the target (fish group) in a circle along the ocean floor, moving its tail up and down near the sand to form a ring or plume of mud. It causes the fish to become confused and jump out of the water to the mouths of the dolphins waiting along the outer edge of the mud ring. Bottlenose dolphins' foraging behaviour is simulated by MROA, which begins with the swarm of dolphins utilizing echolocation to find prey and ends with the formation of a mud ring for feeding. Prior to mathematically replicating the MRA algorithm, the following points must be described:

The K parameter illustrates how the dolphin swarm consistently approaches the prey at the initial stage of the hunting process. The transition between the mud ring (exploitation) and prey-searching (exploration) processes is controlled by this parameter, which is the decreased sound volume each time the swarm approaches the prey. The MROA algorithm looks everywhere during the exploration phase and looks for superior solutions during the exploitation phase. The exploration process occurs in the search space if this parameter is high; when it is low, it shifts to the exploitation phase.

4. System model and problem formulation

The system model and workflow of the proposed framework are shown in Figs. 1 and 2. Within the MCE, several clouds work together to deliver integrated services to clients via the Internet. Through various cloud service providers, users can carry out the tasks they have submitted. Under a single SLA, these clouds provide the services. Every cloud has a 'manager' server that is aware of its computational resource status. Using virtual machines (VMs) in data centres can make resources available to clients. Every VM has unique features regarding energy usage, storage space, and processing capability. The manager server of a cloud verifies the SLA level of

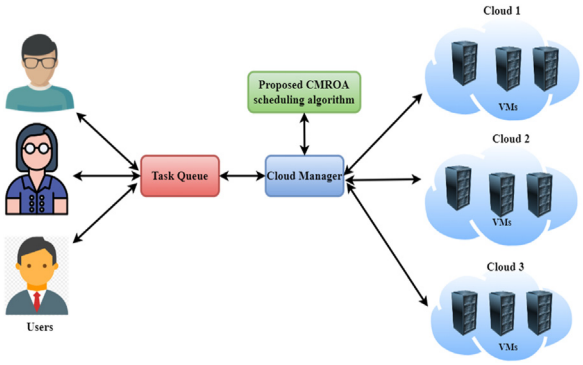

Fig. 1. System model.

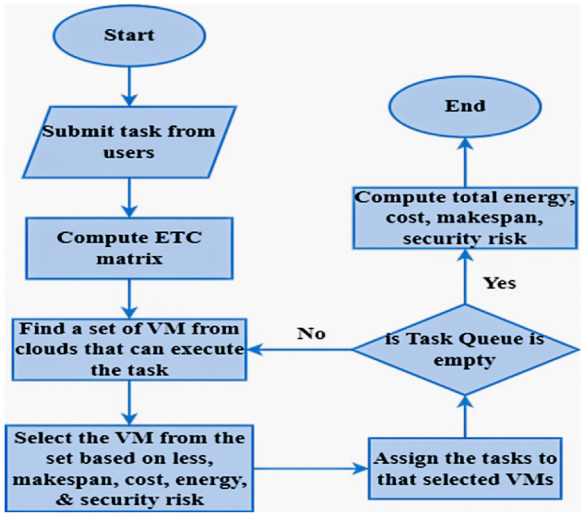

Fig. 2. Workflow of the proposed approach.

each request that comes from a customer. With the assistance of other parties, the server must ascertain the client task's execution time, cost, and energy consumption across all accessible clouds. If the SLA level is met in its own cloud, the job is queued and carried out when the resource is available. The work is moved to another cloud if the SLA is not met. However, the requirements for each work vary because of the different data properties of tasks in the multi-cloud context. The amount of time, money, and energy needed to complete each work varies. Disregarding these variations can lead to an imbalance between the resources allotted to assignments and the resources those activities truly require, particularly in cases where there are notable variations in the data attributes. This discrepancy may result in wasted resources and poor system performance. Therefore, an efficient task-scheduling model is needed to assign user requests to the relevant VMs. Furthermore, integrating risk probability into work scheduling might improve overall reliability and protection in a multi-cloud system. As a result, tasks can be allocated to resources that reduce any security risks and meet their performance needs, guaranteeing a more resistant and robust system.

Let us assume a group of clouds C = {C1, C2, C3, …, Cm}, where each cloud C_k, $1 \le k \le m$ contains a cloud service provider (CSP). Regarding computational resources, each cloud C_k, $1 \le k \le m$ is distinct from other clouds C_t, $t \ne k$, $1 \le k \le m$. Also, take into consideration a group of tasks T = {T1, T2, T3, …, Tl}, where every job Ti, $1 \le i \le l$ is distinct from another job T_j, $i \ne j$, $1 \le j \le l$. Every task Ti is one that a consumer requests from a CSP. It has millions of instructions (MI). In order to complete these tasks, take a look at a set of VMs V = {V1, V2, V3, …, Vn} where |Vn| is the total number of VMs in cloud C_k and a Vj ∈ Vn, $1 \le i \le n$. Note that $N = \sum_{j=1}^{n} |V_j|$ where n is the total number of VMs in all the clouds.

To allocate the task set T to the virtual machine set V (f: T → V) of cloud set C in a way that satisfies the subsequent goals. (1) There is a reduction in the total processing time (makespan). (2) Reduced energy usage; (3) Reduced execution cost; and (4) Reduced possibility of security risk. These goals' explanations are provided below.

4.1. Makespan

Makespan is the amount of time that the multi-cloud network needs to run through all of the input tasks. Let M[k] be a specific cloud's makespan, k, $1 \le k \le m$, and let F [i, k], $1 \le i \le l$, be a Boolean variable with the following definition:

$$F[i,k] = \begin{cases} 1 & \text{if the task } T_i \text{ is assigned to cloud } C_k \\ 0; & \text{otherwise} \end{cases} \quad (1)$$

Then, the following is the mathematical expression for the makespan of a specific cloud Ck:

$$M[k] = \sum_{i=1}^{l} ETC[i,k] \times F[i,k] \quad (2)$$

Here, the expected time to compute matrix is denoted by ETC, which denotes the task's execution time on the cloud. It is represented in the following Equation.

$$ETC = \begin{array}{c} \\ T_1 \\ T_2 \\ \dots \\ T_l \end{array} \begin{array}{c} \begin{array}{cccc} C_1 & C_2 & \dots & C_m \end{array} \\ \begin{pmatrix} ETC_{11} & ETC_{12} & \dots & ETC_{1m} \\ ETC_{21} & ETC_{22} & \dots & ETC_{2m} \\ \dots & \dots & \dots & \dots \\ ETC_{l1} & ETC_{l2} & \dots & ETC_{lm} \end{pmatrix} \end{array} \quad (3)$$

Here, the execution time required to complete task Ti on cloud Ck is indicated by ETC_{ik}, $1 \le i \le l$, and $1 \le k \le m$. A cloud's processing speed (Ck) measured in a million instructions per second (MIPS) divided by the task Ti's instruction length (MI) is called an element ETC_{ik}.

As a result, the overall makespan is shown as follows,

$$Mkspan = max(M[k]), 1 \le k \le m \quad (4)$$

4.2. Cost

Storage, transport, and Computation costs are all included in the task completion cost. Task T_i on virtual machine Vj has a computational cost that depends on task size (TCi) and task computation unit cost (Pj). The transfer cost is connected to the data size (TLi) that needs to be transmitted to complete the job and the per-unit transfer cost (Qj) on the Vj. After the data necessary for the task has been transferred, storage expenses arise because the data needs to be stored. Storage expenses depend on the amount of data that has to be delivered to complete the task (TLi) and the cost per unit for storing data on the virtual machine (Vj) (Sj). As a result, the following represents the task Ti's execution cost on the Vj:

$$Cost(T_i, V_{jk}) = TC_i \times P_{jk} + TL_i \times Q_{jk} + TL_i \times S_{jk} \quad (5)$$

Where $TC_i \times P_{jk}$ represents the computational cost needed to complete task Ti on the jth virtual machine in the kth cloud, $TL_i \times Q_{jk}$ represents the transfer cost needed to complete task Ti on the jth virtual machine in the kth cloud, and $TL_i \times S_{jk}$ represents the storage cost needed to complete task Ti on the jth virtual machine in the kth cloud, The task execution cost should be less than the budgeted cost TMi, or cost $\leq$ TMi, subject to the cost budget. Thus, the following is an expression for the overall job completion cost:

$$T\,cos\,t = \sum_{k=1}^{m}\sum_{j=1}^{n}\sum_{i=1}^{l} Cost\left(T_i, V_{jk}\right) \tag{6}$$

4.3. Energy consumption

The energy utilization approach is essential in a multi-cloud context because it shows how cloud service energy consumption impacts overall operational expenses and environmental issues like energy conservation and reducing emissions from cloud platforms. The task execution time, task size (TCi) and CPU computing power (Cj) generated per unit time are related to the energy consumption of task Ti when it is executed on the resource Vj. The energy consumption of the task Ti running on resource Vj in the kth cloud is thus expressed as follows:

$$EC\left(T_i, V_{jk}\right) = \frac{TC_i}{C_j} \times V_{jk} \tag{7}$$

After that, the overall energy usage can be stated as

$$TEC = \sum_{i=1}^{l}\sum_{j=1}^{n}\sum_{k=1}^{m} EC\left(T_i, V_{jk}\right) \tag{8}$$

4.4. Security risk probability

Another crucial concern is security, which is considered to be the biggest obstacle facing cloud computing. The scheduler attempts to provide security and save energy costs simultaneously. This setting balances the virtual machine's security level and the likelihood of task danger. Three security services are employed to protect the tasks: confidentiality, integrity, and authentication. Users who dynamically integrate these security services can attain adequate protection against various hazards and attacks. A distinct set of security levels falls under each of these categories, and each one is linked to a normalized performance rating that ranges from 0.1 to 1.0. Each service category's greatest performance value is 1.0, and its lowest normalized performance value is 0.1. The suggested model states that a typical activity will include three distinct security services with various user-defined security rates. The risk probability calculated for a specific task is determined using Equation (9). It aids in determining each task-VM pairing's security compatibility, ensuring that tasks requiring a high level of security are not allocated to VMs that cannot meet these requirements.

$$P\left(T_i^q, V_j^q\right) = \begin{cases} 0 & if\ SD_i^q \leq SS_j^q \\ 1 - e^{-\left(SD_i^q \leq SS_j^q\right)} & otherwise \end{cases} \quad q \in a, b, c \tag{9}$$

Here, a, b, and c denote the authentication, integrity and confidentiality, respectively, SS_j^q denote the security services virtual machine V_j, and SD_i^q denote the security demand of task T_i.

In this, the probability of a security risk for a task on a VM is zero if the task's security demand (SD_i^q) is less than or equal to the VM's security capability (SS_j^q). It suggests that no perceived danger exists because the VM can manage the task's security requirements. However, if the security need of the job is greater than the security service provided by the VM, the probability of risk is $1 - e^{-(SD_i^q \leq SS_j^q)}$, which indicates that risk increases exponentially when the variance between security demand and VM capacity increases.

The total risk probability for each activity across all virtual machines was calculated by

$$P\left(T_i, V_j\right) = 1 - \prod_{i \in T, j \in n}\left(1 - P\left(T_i^q, V_j^q\right)\right) \tag{10}$$

In a multi-cloud scenario, the total risk probability is calculated by

$$ORP = 1 - \prod_{i \in T, j \in n, k \in m}\left(1 - P\left(T_i, V_{jk}\right)\right) \tag{11}$$

This constraint maximizes the system's capacity to reduce security flaws while tasks are being executed by allowing for a thorough and detailed evaluation of security risks in task scheduling.

4.5. Main objective function

This section contains the work's primary goal. The following Equation represents the single objective we created by combining the previously mentioned four multi-objectives. It is employed to determine the job scheduling strategy's fitness function.

$$F = min\left(\left(W1 \times Mkspan\right) + \left(W2 \times TCost\right) + \left(W3 \times TEC\right) + \left(W4 \times ORP\right)\right) \quad (12)$$

The variables Mkspan, TCost, TEC, and ORP are makespan, task utilization cost, energy consumption, and probability of security risk. The fuzzy triangular membership function calculates W1, W2, W3, and W4 weights. In Eq. (13), the weight computation is explained.

$$W = \begin{cases} 0 & if\ r < f \\ \dfrac{r-f}{p-f} & if\ f \le r \le f \\ \dfrac{q-r}{q-p} & if\ p \le r \le q \\ 0 & if\ r \ge q \end{cases} \quad (13)$$

Here, the vertices of the triangle membership function T (f) are denoted by p, q, and r. There are three boundaries: r is the upper boundary with a membership value of 0, q is the medium border with a membership value of 1, and p is the lower boundary.

5. Proposed hybrid scheduling algorithm

The new hybrid multi-objective task scheduling method known as CMROA in a multi-cloud context is presented in this section. Energy consumption, makespan, cost-effectiveness, and security risk probability are all considered as its objectives. This hybrid algorithm integrates the exploration phase of the Crayfish Optimization Algorithm (COA) and the Exploitation phase of the Mud Ring Optimization Algorithm (MROA) to overcome the drawbacks of the conventional COA approach, such as the unequal capacity for exploration and exploitation, vulnerability to premature optimization, and a tendency toward stagnation. To improve the exploitive behaviours of COA, we integrate the exploitation operators from the MROA to overcome these flaws. MROA simulates the feeding behaviour of bottlenose dolphins. Applying MROA movement patterns can aid in local search and solution optimization.

Consequently, the CMROA algorithm combines the advantages of COA's exploring capability with MROA's exploitation procedure and finds the best solution. In order to increase diversity, COA first uses its exploration behaviour to examine the solution space thoroughly. After identifying possible solutions, MROA evaluates them using its exploitation behaviour to determine the Pareto-optimal answers.

5.1. Initialization

To establish an appropriate foundation for the ensuing optimization process, the population-based algorithm CMROA begins with population initialization. Each crayfish's position in the CMROA modelling indicates a possible answer to an issue with a D dimension, and the population location of N crayfish indicates a collection of potential solutions X, which is displayed as Equation (14).

$$X = \begin{bmatrix} X_1 \\ \vdots \\ X_i \\ \vdots \\ X_N \end{bmatrix} = \begin{bmatrix} X_{1,1} & \cdots & X_{1,j} & \cdots & X_{1,D} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ X_{i,1} & \cdots & X_{i,j} & \cdots & X_{i,D} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ X_{N,1} & \cdots & X_{N,j} & \cdots & X_{N,D} \end{bmatrix}_{N \times D} \quad (14)$$

Here, the problem dimension is denoted by D, the number of crayfish population is denoted by N, and the initial location of the ith crayfish in the jth dimension is denoted by $X_{i,\ j}$. It is generated randomly in the problem's search space; Equation (15) illustrates the specific expression of Xi, j.

$$X_{i,j} = lb_j + \left(ub_j - lb_j\right).r, i = 1, 2, \cdots, N; j = 1, 2, \cdots, D \quad (15)$$

Here, r indicates the uniformly distributed random number belonging to [0, 1]; the upper and lower bound is denoted by ub_j and lb_j.

Crayfish go through distinct stages depending on the surrounding temperature. Temperature has an impact on their food intake as well, and it varies roughly with temperature. The definition of temperature in this algorithm is given by equation (16).

$$Temp = 20 + r.15 \quad (16)$$

Here, Temp denotes the outside temperature. Equation (17) provides the mathematical representation of the crayfish's food intake, or p.

$$p = C_1 \cdot \left(\frac{1}{\sqrt{2\pi}.\sigma} \cdot exp\left(-\frac{\left(Temp - \mu\right)^2}{2\sigma^2}\right)\right) \quad (17)$$

Here, C_1 and σ are used to regulate the amount of food that crayfish consume at various ambient temperatures, and μ is the ideal temperature.

5.2. Exploration

A crayfish will select the heat escape stage—th X_{shade} cave—when the temperature rises above 30 °C. Equation (18) illustrates the definition of cave in mathematics.

$$X_{shade} = 0.5 \cdot (X_G + X_L) \quad (18)$$

Here, X_L is the ideal spot for the present crayfish population, and X_G is the ideal location reached by the algorithm iteration thus far. There can be conflict among crawfish when entering the heated area. If there are numerous crayfish and insufficient holes, they will fight for the same spot to hide from the heat. If there are more caverns, this will not be the case. The algorithm used a random number between 0 and 1 called rand to figure out whether a competition has occurred. There is no competition for the cave, and crayfish can enter immediately to cool off when the random number rand<0.5. Equation (19) displays this process's mathematical expression.

$$X_{i,j}^{t+1} = X_{i,j}^{t} + C_2 \cdot r \cdot \left(X_{shade} - X_{i,j}^{t}\right) \quad (19)$$

In this case, the random number between 0 and 1 is denoted by r, the current number of iterations is denoted by t, and $t+1$ denotes the number of iterations of the subsequent generation. Moreover, the current position of the ith crayfish in the jth dimension is denoted by $X_{i,j}^{t}$, and the value of $C2$ decreased when the number of iterations is increased, which is expressed as,

$$C_2 = 2 - \left(\frac{t}{T}\right), t = 1, 2, \cdots, T \quad (20)$$

Here, the maximum and current iteration is denoted by T and t correspondingly, the shaded cave's position is denoted by X_{shade}, which is computed as the mean of optimum within the current crayfish swarm X_L and the current best solution X_G. When the temperature rises above 30 °C, and the random number rand is more significant than 0.5, several crayfish will fight for a cave and move on to the competition stage. At this point, Equation (21) displays the revised position of the crayfish.

$$X_{i,j}^{t+1} = X_{i,j}^{t} - X_{z,j}^{t} + X_{shade} \quad (21)$$

Here, the random crayfish in the population is denoted by z, defined in the Equation below.

$$z = round(r.(N-1)) + 1 \quad (22)$$

Here, round is the integer function, and the random number from 1 to 0 is denoted as r.

5.3. Exploitation

Once out of the heat, the crayfish will search for food. To find the best answers during this exploitation phase, we use MROA's exploitation behaviour. In this section, dolphins can find and encircle the prey after spotting it. Since the location of the optimal design in the search space is unknown beforehand, the MRA technique regards the target prey (optimal or close to it) as the best solution available. Once the best search agent has been identified, the other dolphins will try to adjust their locations based on the best dolphin position. The following equations explain this behaviour:

$$\overrightarrow{A} = \left| \overrightarrow{C}\overrightarrow{D}^{*t-1} - \overrightarrow{D}^{t-1} \right| \quad (23)$$

$$\overrightarrow{D} = \overrightarrow{D}^{*t-1} \cdot sin(2\pi l) - \overrightarrow{K} \cdot \overrightarrow{A} \quad (24)$$

Here, $\overrightarrow{D}^{*}$ is the position vector of the best dolphin position to date, $\overrightarrow{D}$ is the dolphin position vector, and t denotes the current time step, l is a random number. Also $\overrightarrow{,K}$ and $\overrightarrow{C}$ are coefficient vectors. If there is an improved position, the monitoring $\overrightarrow{D}$ should be changed at each time step. As the other dolphins surround the prey, we will notice that the best dolphin circles around the object while quickly sweeping its tail across the beach to form a sine wave and make a plume.

The calculation of the vector $\overrightarrow{C}$ is as follows:

$$\overrightarrow{C} = 2 \cdot \overrightarrow{r}$$

Any place within the search area can be reached by figuring out the random vector $\overrightarrow{r}$. As a result, Eq. (24) mimics the surrounding prey and aids any dolphin in defending a place near the optimal one. The revised value—the optimal fitness value—is used to assign the tasks. The Flowchart of the proposed CMROA algorithm is given in Fig. 3, and pseudocode is given in Algorithm 1.

Algorithm 1

Start
Initialize swarm size (number of tasks), maximum iterations T, each task with a random assignment to available cloud resources (VMs), and other algorithm-specific parameters.
While t<T
Defining temperature temp by Eq.(16).
If `temp > 30` then
Execute the exploitation phase of MROA.
Create a "mud ring" by collecting tasks to investigate locally optimal allocations.
Update the task allocation based on Equation (24).
Else
Execute the exploration phase of COA.
If `rand > 0.5` then
Use Equation (21) to simulate competition among tasks, aiming to find a suitable VM.
Else
Use Equation (19) to enter new VMs and explore alternative allocations.
End if
End if
Update the fitness of each task
t=t+1
End While
Deliver the optimal scheduling solution
End

6. Simulation and results

The simulation and the suggested scheduler's outcomes are covered in this section. All trials were conducted on a Windows 10 machine with an Intel Core i5 processor running at 2.40 GHz, 4 GB of RAM, and the CloudSim 3.0.3 toolkit. For performance evaluation, the suggested method used a variety of real-time supercomputing workloads, including HPC2N and NASA. There were 100 iterations throughout the entire simulation. Based on several parameters, such as makespan, cost, energy consumption, and risk assessment, the suggested approach was finally compared to current approaches like the conventional COA [33], conventional MROA [34], MOPTSA3c [22], and MOTSWAO [35] algorithms. Table 2 displays the simulation and configuration settings utilized in the suggested scheduler. Three distinct clouds, which varied mainly in execution speed and cost, were used in the simulations to represent multi-cloud features. Although it operates more slowly, Cloud 1 is more cost-effective for simple operations. Cloud 2 offers moderate pricing and speed. Cloud 3 is more expensive but speedier; it works well for bigger jobs. Moreover, the suggested CMROA algorithm's fine-tuned parameters are shown in Table 3.

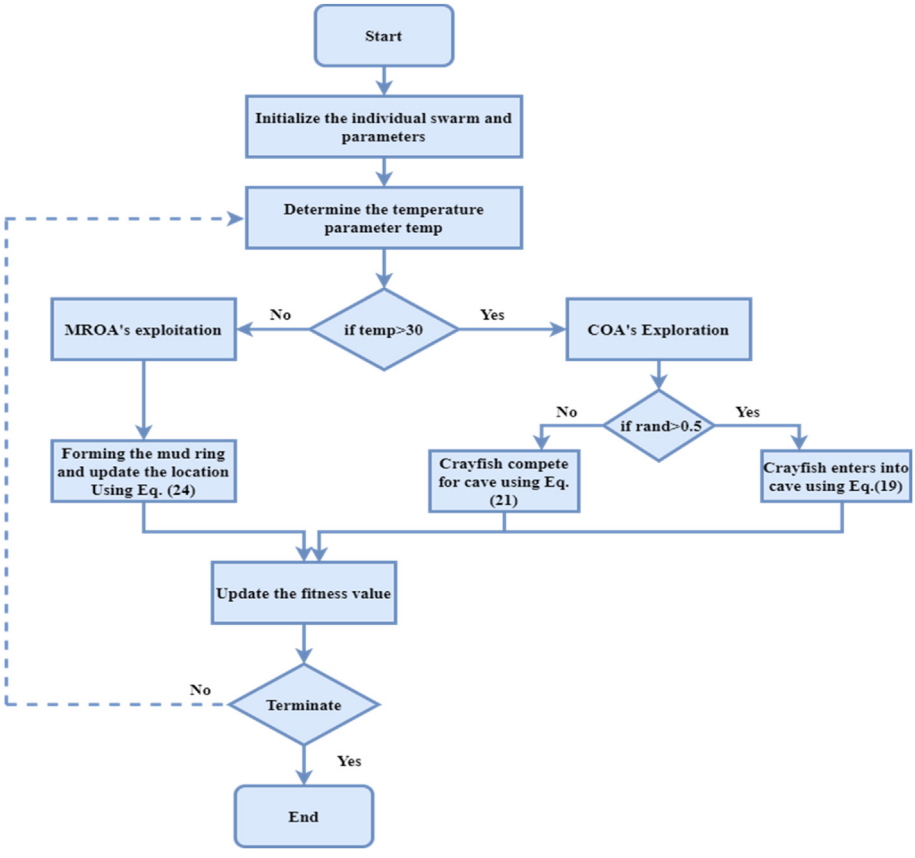

Fig. 3. Flowchart of the CMROA algorithm.

Table 2. Simulation parameters.

Entity	Quantity
Cloud	3
VMs	1000
PM's Memory	32–64 GB
PM's bandwidth	1200 Mbps
Storage of PM	4-8 TB
Memory of VM	4–8 GB
Storage of VM	32–64 GB
VM's Bandwidth	10–30 Mbps
Data centers	70

Table 3. The proposed algorithm's parameters.

Parameter	Value
C1	0.2
μ	25
σ	3
$\vec{r}$	[0,1]

6.1. Workloads

In the studies, two real datasets were used to evaluate the efficacy of the proposed CMROA algorithm. They are High-Performance Computing Center North (HPC2N) and NASA Ames iPSC/860 workloads. These parallel workloads are supplied to the educational sector in the conventional workload style. The cleaned versions of the NASA iPSC and HPC2N logs were used in this experiment. Ake Sandgren, Bill Nitzberg, and Victor Hazlewood are the providers of these workload archives, available in the standard workload format (.swf) that the CloudSim program can detect. NASA has statistics on 42,264 tasks, while HPC2N has statistics on 527,371 tasks. The characteristics of these workloads are given in Table 4. These workloads are selected because they closely resemble a variety of complex, resource-intensive jobs that are frequently managed in multi-cloud settings. These workloads are perfect for evaluating task scheduling algorithms like CMROA because they are dynamic, providing a range of computing tasks, data processing requirements, and network bandwidth requirements. Task diversity aids in evaluating CMROA's resilience to handling the varied and homogeneous workloads typical in real cloud settings.

6.2. Performance metrics

In the multi-cloud computing context, the suggested job scheduling method was evaluated using four performance criteria. The performance parameters determined are makespan, energy consumption, cost, and possibility of security risk. These criteria, the most commonly used assessment metrics, can be used to compare and analyze the efficacy of work scheduling strategies in cloud computing. Equations (4), (6), (8) and (11) are used to calculate the values of these performance metrics, as mentioned in Section 3.

Table 4. Characteristics of the workloads.

Workload	Period	Months	Users	Tasks	CPUs
HPC2N	Jul 02–Jan 06	42	257	52737′	240
NASA	Oct 93–Dec 93	3	69	42264	128

6.3. Analysis results for HPC2N workloads

The evaluation of makespan for the suggested scheduling technique on HPC2N workloads is covered in this subsection. Makespan should be evaluated since it directly impacts the cloud paradigm's scheduling mechanism. An ineffective task scheduler lengthens the makespan, which affects the cloud service provider's QoS. It encourages us to use various real-time workloads to assess the makespan of the CMROA scheduler in a multi-cloud scenario. The estimated makespan for CMROA utilizing the HPC2N workload is shown in Fig. 4 and Table 5. In order to assess the effectiveness of CMROA in light of makespan, the results of our suggested CMROA are compared to those of other algorithms, including COA, MROA, MOPTSA3c, and MOTSWAO. We examined between 200 and 1200 tasks to determine makespan. With 200, 400, 600, 800, 1000, and 1200 tasks, the best makespan that was generated for CMROA is 687.92, 834.71, 1037.86, 1275.39, 1421.56, and 1612.92, in that order. It demonstrates that CMROA learns the policies the scheduler gives even when the number of tasks is increased from 200 to 12000, outperforming all other methods by minimizing makespan for HPC2N workloads.

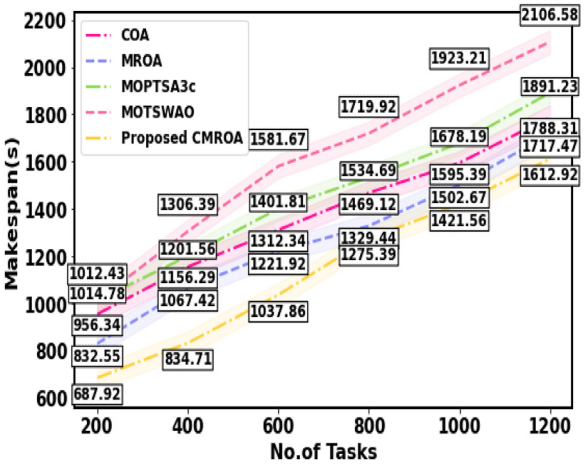

Fig. 4. Analysis of Makespan (best) on HPC2N workload.

Table 5. Makespan analysis on HPC2N workload.

Number of Tasks	Statistics	COA [33]	MROA [34]	MOPTSA3c [22]	MOTSWAO [35]	Proposed CMROA
200	Best	956.34	832.55	1014.78	1012.43	687.92
	Average	983.39	869.29	1093.39	1167.45	693.67
	Worst	998.17	882.81	1112.77	1217.08	707.81
400	Best	1156.29	1067.42	1201.56	1306.39	834.71
	Average	1193.36	1086.23	1279.22	1337.23	856.68
	Worst	1204.61	1106.39	1302.37	1397.29	893.67
600	Best	1312.34	1221.92	1401.81	1581.67	1037.86
	Average	1345.19	1244.55	1428.19	1604.45	1079.66
	Worst	1376.91	1251.65	1449.63	1618.34	1102.82
800	Best	1469.12	1329.44	1534.69	1719.92	1275.39
	Average	1483.44	1331.77	1559.27	1736.76	1289.71
	Worst	1491.03	1393.64	1600.06	1778.90	1296.41
1000	Best	1595.39	1502.67	1678.19	1923.21	1421.56
	Average	1616.52	1586.12	1695.28	1977.23	1448.44
	Worst	1662.21	1591.69	1713.44	2003.56	1489.04
1200	Best	1788.31	1717.47	1891.23	2106.58	1612.92
	Average	1802.98	1769.53	1936.03	2259.53	1662.61
	Worst	1836.35	1812.64	1982.19	2306.49	1669.25

The suggested CMROA performed better than the others, with a Makespan of just 687.92 ms for 200 tasks, compared to 956.34 ms for COA, 832.55 ms for MROA, 1014.78 ms for MOPTSA3c, and 1012.43 ms for MOTSWAO. CMROA consistently performed better than the other models when the number of tasks was increased. CMROA's unique approach to task scheduling—combining COA and MROA—is the cause for its exceptional performance in makespan. These bioinspired algorithms successfully handle challenging optimization issues and effectively schedule jobs by mimicking natural processes.

Following the computation of makespan, we computed energy consumption for our CMROA scheduler. It is a necessary step in the scheduler design process because energy consumption is a crucial metric from the standpoints of the service provider and cloud consumer. Energy consumption reduction helps the cloud provider by resulting in significant power bills, helps cloud users by enabling them to use cloud services at a lower cost, and helps the environment overall. For this reason, we simulated 100 iterations and determined Energy Consumption by assigning 200−1200 jobs. The outcomes are shown in Fig. 5. It demonstrates that, for the HPC2N dataset, the suggested CMROA method achieves the lowest energy consumption compared to other peer algorithms. Furthermore, the results showed that the suggested method leads to near-optimal solutions by offering improved quality and variety.

Next, using the HPC2N dataset, we assess the cost performance indicator. Cost is the total amount generated according to resource usage or utilization that cloud users pay to cloud providers. The main

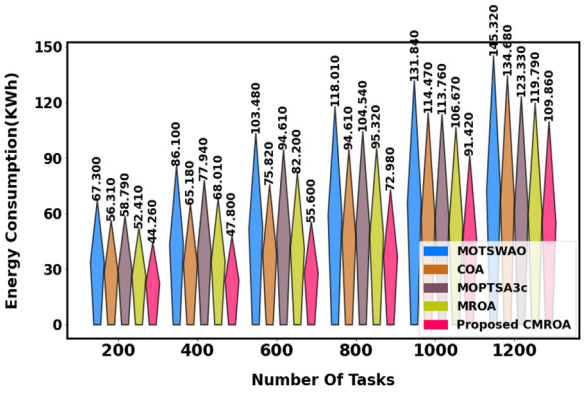

Fig. 5. Analysis of energy consumption on HPC2N workload.

goal is to lower cloud users' costs while increasing cloud providers' profits and revenue growth through efficiently utilizing resources. The cost of scheduling algorithms rises as the number of activities increases, as seen in Fig. 6. When the tasks rise, the CMROA algorithm's computation cost is less than that of the other four algorithms. This result is achieved due to its ability to transition between the phases of exploration and exploitation adaptively; it guarantees a well-balanced search process that prevents early convergence and preserves a high degree of solution variety. As a result, the suggested CMROA algorithm would assist cloud providers in boosting revenue and profit while utilizing the cloud computing environment, as well as cloud users in lowering costs.

Subsequently, we assess the security risk probability measure, which characterizes the security risk of approaches by considering the risk of task scheduling on virtual machines. The chance of risk

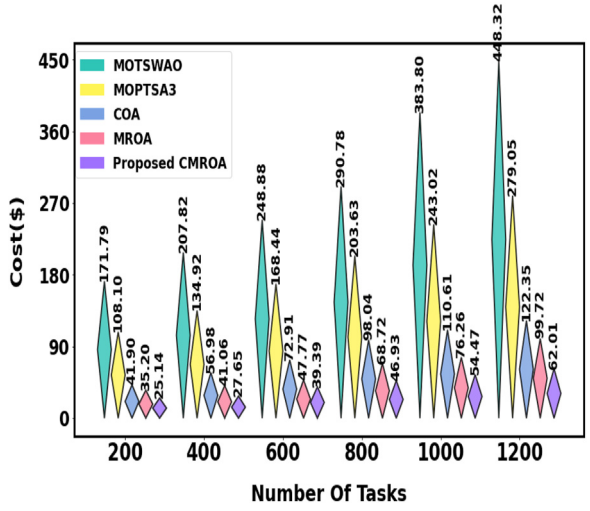

Fig. 6. Analysis of cost on HPC2N workload.

is zero when all security requirements for a task are met. During the comparative analysis process, we identified that the security risk when utilizing the proposed approach is minimal compared to other approaches, as stated in Fig. 7. In the proposed CMROA method, the COA's exploration capabilities guarantee that a wide range of viable alternatives are considered, and the MROA exploitation phase further refines these solutions. This process lowers the probability of security threats resulting from inconsistent or inefficient job scheduling and guarantees that the chosen schedules not only maximize efficiency but also reduce security threats.

In contrast, MOPTSA3C and MOTSWAO are 64.29 % and 61.54 % less effective because they lack a straightforward risk-reduction approach. It is also evident that the CMROA algorithm outperforms MORA by 60.78 %. It is so that the task's security level and the assigned virtual machine's capabilities are in line. CMROA ensures that tasks with high-security requirements are assigned to virtual machines that provide enhanced security services.

Fig. 8 displays the convergence study of the suggested CMROA-based task scheduling in a multi-cloud context compared to the traditional methods. In the seventy-eighth iteration, the proposed CMROA converges first, with a minor deviation, to the MROA. MROA, COA, MOTSWAO, and MOPTSA3C followed. In determining the best solution for case 1, the suggested CMROA improved by 63.32 % compared to MOPTSA3C, 52.43 % compared to MOTSWAO, 34.21 % compared to MROA, and 45.17 % compared to COA during the 50th iteration. Based on the comprehensive examination, it can be inferred that the suggested CMROA finds the optimal solution and effectively schedules the task more quickly than other traditional algorithms.

6.4. Evaluation results for NASA workloads

The evaluation of the results and the discussion of the proposed CMROA job scheduling strategy for NASA workload are presented in this section. As seen in Fig. 9, we generated the graph for the best solution (i.e., best makespan) versus the number of tasks for the NASA dataset to illustrate the performance of CMROA against COA, MROA, MOPTSA3c, and MOTSWAO algorithms. According to Table 6's reported results for this performance indicator, CMROA typically finds a better average makespan than the other scheduling algorithms under evaluation. It indicates that the CMROA operates faster than all other scheduling algorithms in every test case and requires a shorter period to complete the submitted tasks.

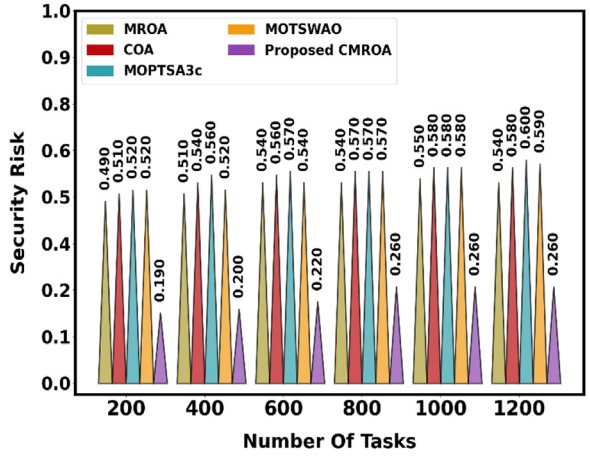

Fig. 7. Security risk analysis on HPC2N workload.

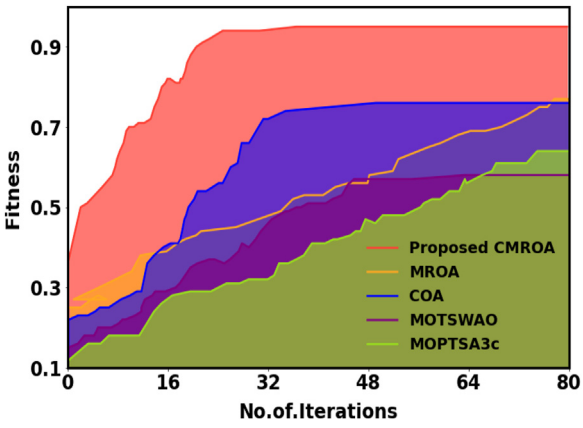

Fig. 8. Convergence analysis on HPC2N workload.

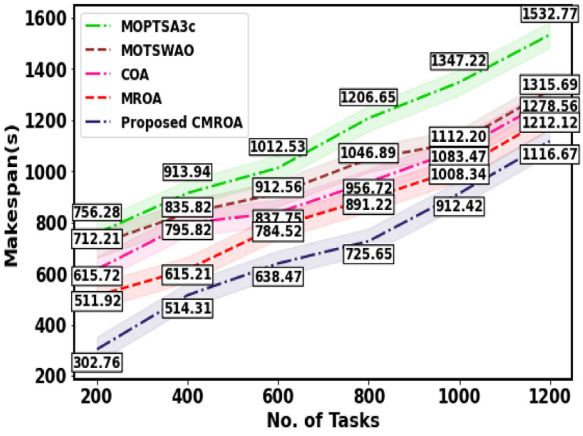

Fig. 9. Analysis of Makespan (best values) on NASA workload.

To be more precise, the outcomes show that the MROA comes in second. Additionally, we discover that MOTSWAO generally outperforms MOPTSA3c, yet both algorithms lag behind the CMROA algorithm. It demonstrates that the average makespan values obtained by applying CMROA are more competitive than those obtained by the other scheduling strategies that were assessed. It means that the suggested CMROA algorithm would help cloud customers save more money while utilizing the multi-cloud environment. Additionally, it minimizes service interruptions and increases user satisfaction because jobs are consistently done on time. In addition, timely completion of tasks is essential for applications with stringent time constraints, like money transactions or real-time data processing. CMROA is an excellent option for various cloud-based applications because of its consistent ability to attain low makespan values, which enhances cloud computing's overall reliability and QoS.

The results of the suggested approach and alternative methods' energy consumption (kWh) are displayed in Fig. 10 (NASA iPSC dataset). As demonstrated in Fig. 10, the suggested approach (CMROA) releases less CO2 and uses less energy to do the necessary tasks than alternative methods (COA, MROA, MOPTSA3c, and MOTSWAO). It shows that COA, MROA, MOPTSA3c, MOTSWAO, and CMROA use more energy when the number of jobs increases from 200 to 1200. However, the suggested algorithm uses less energy for all work sizes than previous methods. Since it efficiently distributes the workload across the available VMs, guaranteeing the best possible use of resources. It avoids overloading some VMs while underutilizing others,

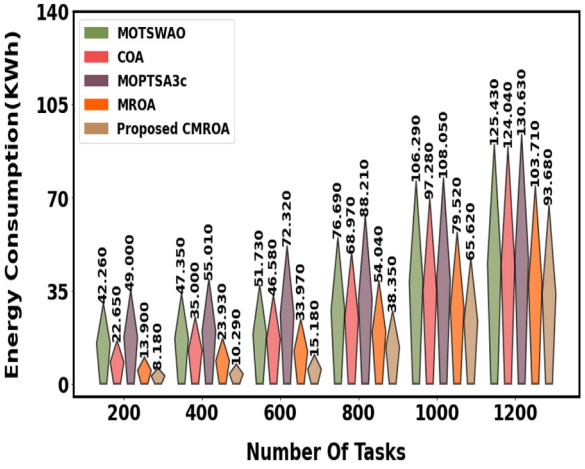

Fig. 10. Analysis of energy consumption on NASA workload.

Table 6. Makespan evaluation on NASA workload.

Number of Tasks	Statistics	COA [33]	MROAv [34]	MOPTSA3c [22]	MOTSWAO [35]	CMROA
200	Best	615.72	511.92	756.28	712.21	302.76
	Average	651.55	537.39	788.01	722.89	326.85
	Worst	670.42	549.65	803.56	745.80	394.11
400	Best	795.82	615.21	913.94	835.82	514.31
	Average	826.00	625.84	926.61	846.72	533.87
	Worst	839.28	638.90	941.27	862.91	554.93
600	Best	837.75	784.52	1012.53	912.56	638.47
	Average	841.03	796.42	1047.74	935.13	647.62
	Worst	850.33	803.78	1073.97	939.62	651.48
800	Best	956.72	891.22	1206.65	1046.89	725.65
	Average	973.04	908.45	1227.40	1067.31	746.27
	Worst	989.94	912.67	1243.67	1084.55	753.06
1000	Best	1083.47	1008.34	1347.22	1112.20	912.42
	Average	1104.83	1024.67	1374.46	1159.83	948.35
	Worst	1125.73	1035.78	1383.22	1169.71	969.35
1200	Best	1278.56	1212.12	1532.77	1315.69	1116.67
	Average	1291.23	1243.76	1569.07	1357.03	1157.73
	Worst	1306.21	1257.81	1581.96	1382.31	1193.12

resulting in more energy-efficient utilization. CMROA keeps a close eye on the workload of VMs and allocates jobs to those that can effectively do more work or have the least amount of idle time. This efficient allocation reduces the system's overall energy usage by ensuring VMs do not stay active without aiding in job execution.

Following the energy usage, we discussed the evaluation of resource costs using our suggested CMROA. An efficient scheduler selects appropriate virtual machines (VMs) to build optimized schedules when reducing resource costs. It is the rationale behind assessing resource cost in scheduling in a multi-cloud context. Inefficient scheduling raises resource costs, which is expensive for cloud users and CSPs alike. It encourages us to use CMROA to reduce the cost in multi-cloud environments. Using various real-time workloads, the proposed method is compared to baseline methodologies that are currently in use. Fig. 11 compares the cost for each scheduling strategy using the NASA workload. As we can see from the figure, the suggested method outperformed the other algorithms in terms of cost.

For NASA workloads with sizes between 200 and 1200, the suggested CMROA achieves 69.04 %, 66.78 %, 63.53 %, and 46.01 % better cost reduction than the MOPTSA3c, MOTSWAO, COA, and MROA algorithms. As was previously said, CMROA's task scheduling reduces energy usage. Decreased energy consumption leads directly to lower operating expenses because energy prices account for a large amount of the total cost of operating cloud data centres. Through equitable load distribution, CMROA shields individual virtual machines from extreme pressure, possibly increasing their lifespan and lowering the need for expensive hardware repairs or exchanges.

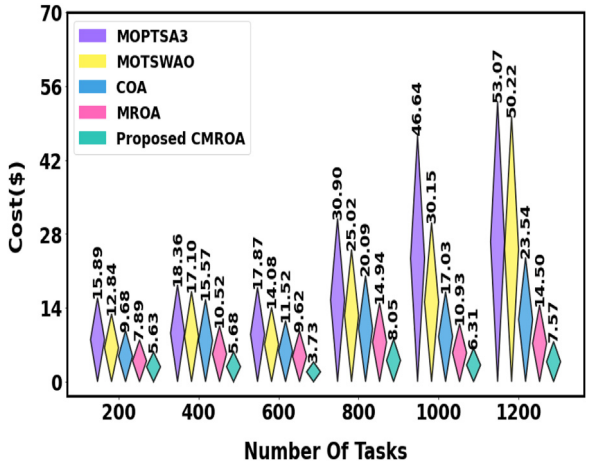

Fig. 11. Analysis of cost on NASA workload.

Fig. 12 shows the security analysis of the proposed and the current methods. The results show that the CMROA balances superior security and lower risk. With 200 tasks in the NASA workload, the CMROA achieves 0.15. The current model values are COA = 0.42, MROA = 0.35, MOPTSA3c = 0.59, and MOTSWAO = 0.51. The recommended method exhibits a lower risk probability than the previous scheduling models when all other tasks are observed. The analysis demonstrates that the proposed model can securely handle task scheduling and execution. The algorithm dynamically modifies work scheduling decisions based on real-time security risk evaluations to ensure the final schedule is resilient against possible threats.

The convergence curves for the different iterations of the NASA workload are shown in Fig. 13. All algorithms have an increasing fitness graph, as can be seen. In other words, every work scheduling scheme raises the baseline population's fitness. Primarily, the convergence curves highlight the highly faster

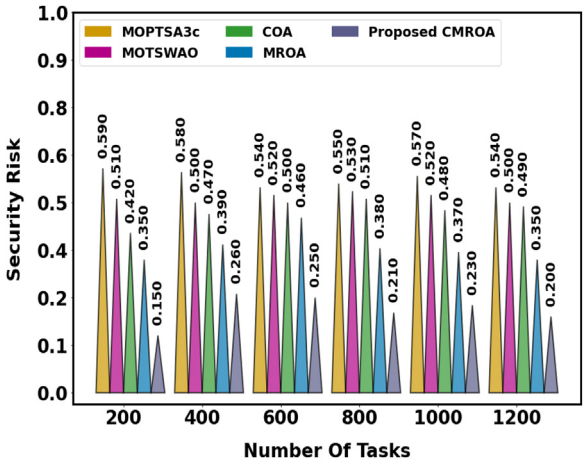

Fig. 12. Security risk evaluation on NASA workload.

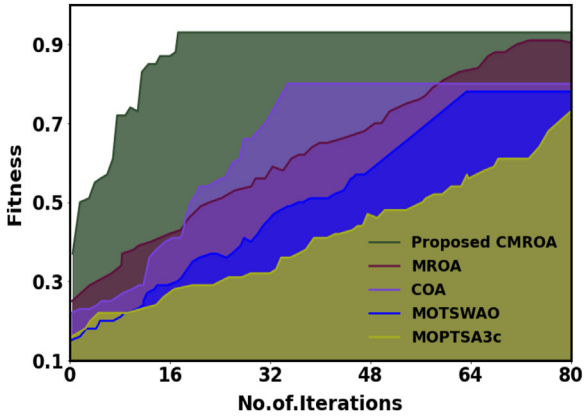

Fig. 13. Convergence analysis on NASA workload.

tendency of the suggested method (CMROA). Because CMROA strongly emphasizes local search, the convergence to the best solution at the end of the iteration is greatly accelerated. Moreover, the tasks are first mapped to the regions to facilitate their fastest possible access to the necessary files for execution. Next, VMs are assigned jobs based on cost, energy, makespan, and security risk. As shown in Fig. 13, the suggested approach outperforms COA, MROA, MOPTSA3c and MOTSWAO by 44.3 %, 33.3 %, 66.3 % and 57.2 %, respectively, and increases fitness.

As a result of the simulation results, we determined that our CMROA solution outperforms the other four approaches regarding job scheduling in a multi-cloud ecosystem. In multi-cloud environments, CMROA is an effective solution for handling job scheduling problems. Its rapid mobility between the exploration and exploitation stages and its ability to achieve a fair balance between them accounts for CMROA's exceptional performance in both domains. Because of its enormous power during the exploitation phase, it may thus acquire the best possible global response and more effectively explore the whole search space. Thus, CMROA is an excellent technique for resolving task-scheduling issues because of all these beneficial characteristics.

6.5. Statistical analysis

Statistical analysis was performed to verify the significance of the resulting data. The results have been validated using the ANOVA approach. A two-way ANOVA test was conducted to determine if task load influences makespan across both datasets. A null hypothesis (H0) states that all groups' means are equal, while an alternative hypothesis (H1) states that they are not. The two-way ANOVA test results are shown in Table 7, where α is set at 0.05. The findings indicate that the null hypothesis for both factors is rejected because the p-values for A (rows) and B (columns) are both less than α. As a result, the alternative theories are proven, confirming the significance of the findings. However, for the interaction between Factors A and B, the p-value is greater than α (0.6719), so the null hypothesis for the interaction is not rejected, indicating that the interaction effect is not statistically significant. It indicates that the two datasets have no influence on the other's individual effects on the outcome and are independent of one another regarding the factors' combined effect.

6.6. Discussion

This section examines the advantages of the suggested approach over competing approaches concerning various performance indicators. The trial's results demonstrate how well the suggested scheduler can lower the energy, cost, makespan, and security risk while allocating the jobs to the appropriate VM. This result is the harmonious result of a few essential elements, such as the exploration of COA and the exploitation of MROA. In order to help the algorithm avoid local optima and find various scheduling solutions, the COA-based exploration module in CMROA allows for extensive searches throughout the solution space. On the other hand, the MROA-based exploitation module is developed to target potential areas found during the exploration stage to fine-tune solutions. This feature guarantees that CMROA can refine job assignments to reduce energy and operating costs and converge towards optimal or nearly optimal solutions. CMROA lowers task completion time (makespan) by carefully balancing exploration and exploitation, guaranteeing quicker and more responsive operations, which are necessary for real-time applications. For 200 tasks, it achieved 687.92 ms and 302.76 ms for HPC2N and NASA workloads, which is lower than existing methods. It is perfect for businesses that require cost-effective resource allocation in multi-cloud setups because it also produces significant cost reductions compared to alternative models in both workloads. Environmental objectives are met by CMROA's energy efficiency, which lowers energy costs and environmental effects.

Additionally, its security-aware scheduling increases data protection and minimizes vulnerabilities, a crucial advantage in industries with strict security requirements. Additionally, the statistical analysis with a two-way ANOVA validates the statistical significance of the reported improvements in

Table 7. ANOVA test.

Source	df	Sum of Square (SS)	Mean Square (MS)	F Statistic (df1, df2)	P-value
Rows (A)	1	3121626.056	3121626.056	104.6747 (1,48)	1.192e-13
Columns (B)	5	4380694.986	876138.9972	29.3788 (5,48)	1.58e-13
Interaction AB	5	95098.5676	19019.7135	0.6378 (5,48)	0.6719
Error	48	1431463.95	29822.1656		
Total	59	9028883.56	153031.9247		

security risk, cost, makespan, and energy usage. Finally, these results indicate that the proposed approach is an effective tool for scheduling tasks in a dynamic multi-cloud environment.

6.7. Ablation study

This subsection includes ablation studies that illustrate the effect of our proposed method. MROA-based exploitation and COA-based exploration are two primary modules that comprise the suggested structure. The test results above showed that the recommended method can produce positive results on the NASA and HPC2N workloads. In this section, we carried out the ablation experiment to investigate the effects of each component in the suggested structure in more detail. For the ablation study, we take the overall result of 200–1200 tasks. Table 8 displays the results of the ablation experiment.

Table 8 demonstrates the comparatively considerable significance of MROA-based exploitation and COA-based exploration. All performance indicators significantly decrease if we remove one of those modules because eliminating COA-based exploration could make the algorithm less capable of examining many options and may restrict its capacity to break out from local optima. Furthermore, the algorithm's output may become more erratic when we remove MROA's exploitation. Without it, the algorithm might be investigated widely, but it might be challenging to get accurate, superior results. It might result in less effective job distribution and decreased scheduling effectiveness overall. Finally, it is evident from the ablation tests that each component of the suggested system is essential to achieving the optimal outcome.

7. Implications, practical benefits for practitioners and real-world applications

The results obtained from the task scheduling model based on CMROA have important practical implications. By lowering execution costs, the algorithm lowers operating costs and increases the viability of multi-cloud solutions. Due to this cost reduction, organizations can distribute resources effectively while staying within budgetary limits. By lowering the makespan, CMROA also increases task completion efficiency. It boosts operational speed and responsiveness, which is essential for real-time applications like online business, statistical analysis, and monetary transactions. Through resource allocation optimization, the approach also reduces energy usage, which benefits cloud providers and massive data centres by lessening their environmental effects and lowering energy prices over time. Additionally, CMROA integrates security-aware scheduling to handle security concerns. It helps to reduce the risks of data breaches and illegal access in multi-cloud systems. While handling sensitive data, this guarantees safer operations.

CMROA helps practitioners optimize resource consumption and achieve higher throughput without putting undue strain on individual systems by making the best use of virtual machines and other resources. Its adaptability to diverse service providers and specialized workloads is made possible by its flexibility across different cloud providers and configurations, which is especially useful in hybrid and multi-cloud settings. Through multi-criteria optimization, the approach also facilitates improved decision-making by enabling practitioners to prioritize cost, energy efficiency, makespan, or security, all of which align with corporate objectives.

Due to its scalability, CMROA is perfect for large-scale and enterprise applications, effectively managing virtual machines and huge task volumes. In practical applications, it is particularly advantageous for data-intensive sectors such as e-commerce, healthcare, and finance, where it maximizes resource distribution among cloud nodes. Furthermore, it facilitates high-performance computing (HPC) jobs in research and development, where safe data handling and processing power are essential.

Table 8. Ablation study.

COA based exploration	MROA based exploitation	Makespan	Cost	Energy	Security risk
HPC2N workload					
✔		1960.12	98.34	167.89	0.832
	✔	1815.86	85.19	154.73	0.787
✔	✔	1145.06	42.60	70.32	0.230
NASA workload					
✔		1318.11	10.16	68.77	0.623
	✔	1267.82	9.05	57.05	0.548
✔	✔	701.71	6.16	38.55	0.216

Organizations can attain operational efficiency and meet industry demands for cost-effectiveness, sustainability, and improved security in multi-cloud task scheduling by putting CMROA into practice.

8. Potential integration challenges in a real multi-cloud environment

This section discusses the potential integration challenges of the proposed approach in a real multi-cloud environment. First, it is crucial to ensure compatibility with various cloud platforms because every cloud provider has different architectures, APIs, and frameworks that may influence interoperability. Furthermore, it is critical to resolve any possible problems with data consistency and make sure that data synchronization is dependable across various cloud settings. In a real multi-cloud environment, the third significant problem is putting in place secure access and data management mechanisms that comply with the security criteria of each provider. Although the majority of cloud providers provide auto-scaling capabilities to modify resources in response to demand dynamically, controlling this process can be challenging because different providers have different auto-scaling policies and configurations. For some applications, one cloud might offer smooth auto-scaling, but another might have restrictions or need human assistance when scaling resources. It could cause irregularities in the allocation of resources and cause delays in the completion of tasks. Cloud providers could store data in many jurisdictions, which could cause problems with compliance when managing sensitive data across borders. Tasks requiring private data must be assigned to particular clouds with extra caution to comply with legal requirements (such as GDPR and HIPAA). Moreover, Task scheduling performance may be impacted by the regional differences in network latency and bandwidth between different clouds, particularly for jobs requiring inter-cloud communication. We will examine these integration issues in our upcoming work and create methods to improve the suggested methodology, guaranteeing its efficient implementation and functionality in a real multi-cloud environment.

9. Limitations and future directions

Even if our proposed framework successfully schedules tasks in a multi-cloud context, some issues still need to be fixed. Although CMROA covers the fundamentals of security-aware scheduling, it could not go far enough in meeting compliance requirements in many regulatory contexts. In our future work, we will incorporate any further procedures or customizations required to meet specific compliance or data protection requirements. Furthermore, the current assessment only includes initial real-world testing and is mainly simulation-based. In our future work, we will apply the suggested work in a real-world setting.

Furthermore, the suggested method is only used in multi-cloud environments. In our future work, we will try to implement it in other fog SDN environments. When resources fail during the execution of tasks, the suggested model does not adequately handle fault tolerance or recovery procedures. To guarantee dependable task execution even in the case of resource or network failures, we will use strong fault-tolerance and recovery techniques in our upcoming work. Although makespan and energy consumption are taken into consideration by the algorithm, the environmental impact of the employed cloud resources is not entirely addressed. Future research can focus on creating carbon-aware scheduling techniques that prioritize greener options when scheduling tasks and consider the ecological consequences of cloud providers. Lastly, when jobs are moved between geographically dispersed cloud centres, task migration and dynamic resource allocation may result in higher delay or communication overhead among cloud providers. Future developments might concentrate on edge computing integration, which would lessen reliance on frequent cloud migrations by executing latency-sensitive jobs closer to the user. It is feasible to maximize response time and system effectiveness by incorporating edge resources into the scheduling algorithm.

10. Conclusion

This paper proposes the CMROA technique-based multi-objective scheduling approach in a multi-cloud environment. The suggested framework contains significant contributions that optimize important aspects like makespan, cost, security, and energy usage. The creation of a new hybrid model that combines MROA and COA algorithms to improve task scheduling efficiency is one of the main contributions. Another is security, cost, energy and makespan-aware scheduling mechanism to safeguard sensitive data across multi-cloud platforms, minimize operating expenses, lower overall energy consumption without compromising performance, and scalability and adaptability to handle a variety of workloads across different cloud environments. Two real-world workloads are utilized to evaluate the performance of this suggested

approach. The results of the experiments show that the proposed CMROA technique performed well in terms of makespan, cost, security, and total energy consumption. More precisely, the gathered data demonstrated that the CMROA outperforms traditional COA and MROA approaches and outperforms all comparison algorithms with an average improvement of 46 % makespan, 55 % energy, 61 % cost, and 52 % security risk in all the examined scenarios. In the future, task scheduling in an edge cloud or fog cloud environment may present with other meta-heuristic algorithms. Furthermore, the performance of the CMROA approach can be evaluated further by testing it in a real-world environment.

Ethical statement

This article does not contain any studies with human participants or animals performed by any of the authors.

Funding body

No funding was received to assist with the preparation of this manuscript.

Conflict of interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We declare that this manuscript is original, has not been published before and is not currently being considered for publication elsewhere.

References

[1] A.N. Malti, M. Hakem, B. Benmammar, A new hybrid multi-objective optimization algorithm for task scheduling in cloud systems, Cluster Comput. 27 (2024) 2525–2548, https://doi.org/10.1007/s10586-023-04099-3.
[2] Y. Pachipala, D.B. Dasari, V.V.R.M. Rao, P. Bethapudi, T. Srinivasarao, Workload prioritization and optimal task scheduling in cloud: introduction to hybrid optimization algorithm, Wireless Network 1 (2024) 1–20, https://doi.org/10.1007/s11276-024-03793-3.
[3] P.B. Jawade, S. Ramachandram, Task scheduling in multi-cloud environment via improved optimization theory, Int. J. Wireless Mobile Comput. 27 (2024) 64–77, https://doi.org/10.1504/IJWMC.2024.139671.
[4] V.K. Vasantham, H. Donavalli, Cost minimization approach with hybrid optimization-based task scheduling in Geo-distributed cloud, Aust. J. Electr. Electro. Comput. 1 (2024) 1–13, https://doi.org/10.1080/1448837X.2024.2355004.
[5] M. Karaja, A. Chaabani, A. Azzouz, L. Ben Said, Efficient bi-level multi objective approach for budget-constrained dynamic bag-of-tasks scheduling problem in heterogeneous multi-cloud environment, Appl. Intell. 53 (8) (2023) 9009–9037, https://doi.org/10.1007/s10489-022-03942-1.
[6] M. Zambianco, S. Cretti, D. Siracusa, Cost minimization in multi-cloud systems with runtime microservice Re-orchestration, in: 2024 27th Conf on Inno Clo 1, 2024, pp. 65–72, https://doi.org/10.1109/ICIN60470.2024.10494463.
[7] R. Kaur, D. Anand, U. Kaur, S. Verma, Kavita, S.W. Park, I. H. Ra, An advanced job scheduling algorithmic architecture to reduce energy consumption and CO2 emissions in multi-cloud, Electronics 12 (2023) 1810, https://doi.org/10.3390/electronics12081810.
[8] S.I. Shyla, T.B. Bell, C.J.J. Sheela, Adaptive golden eagle optimization based multi-objective scientific workflow scheduling on multi-cloud environment, Multimed. Tool. Appl. 83 (16) (2024) 47175–47198, https://doi.org/10.1007/s11042-023-17405-3.
[9] S.R. Murugaiyan, D. Chandramohan, T. Vengattaraman, P. Dhavachelvan, A generic privacy breach preventing methodology for cloud-based web service, Int. J. Grid. Util. Comput. 6 (3) (2014) 53–84, https://doi.org/10.4018/ijghpc.2014070104.
[10] S. Mangalampalli, G.R. Karri, S.N. Mohanty, S. Ali, M.I. Khan, D. Abduvalieva, E.A. Ismail, Fault tolerant trust-based task scheduler using Harris Hawks optimization and deep reinforcement learning in multi cloud environment, Sci. Rep. 13 (1) (2023) 19179, https://doi.org/10.1038/s41598-023-46284-9.
[11] T. Zhao, L. Wu, Z. Cui, X. Cai, Collaborative scheduling of multi-cloud distributed multi-cloud tasks based on evolutionary multi-tasking algorithm, Int. Conf. Bioinspired. Comput. Theor. Appl. 1 (2023) 3–13, https://doi.org/10.1007/978-981-97-2272-3_1.
[12] A.A. Mary, Scientific workflow scheduling using adaptive dingo optimization in multi-cloud environment, Int. J. Inf. Technol. 1 (2024) 1–8, https://doi.org/10.1007/s41870-024-01881-3.
[13] X. Chen, Multi-objective optimization task scheduling method based on dynamic programming for multi-cloud environment, in: 2023 4th Int Conf Inf Sci Parallel Distrib Syst 1, 2023, pp. 278–283, https://doi.org/10.1109/ISPDS58840.2023.10235565.
[14] P.B. Jawade, S. Ramachandram, DAGWO based secure task scheduling in multi-cloud environment with risk probability, Multimed. Tool. Appl. 83 (2024) 2527–2550, https://doi.org/10.1007/s11042-023-15687-1.
[15] Y. Hao, J. Cao, Q. Wang, T. Ma, Q. Wang, X. Zhang, Meteorological data layout and task scheduling in a multi-cloud environment, Eng. Appl. Artif. Intell. 126 (2023) 106860, https://doi.org/10.1016/j.engappai.2023.106860.
[16] S. Wang, Y. Duan, Y. Lei, P. Du, Y. Wang, Electricity-cost-aware multi-workflow scheduling in heterogeneous cloud, Computing 1 (2024) 1–27, https://doi.org/10.1007/s00607-024-01264-3.
[17] K. Malathi, R. Anandan, J.F. Vijay, Cloud environment task scheduling optimization of modified genetic algorithm, J. Internet Serv. Inf. Secur. 13 (2023) 34–43, https://doi.org/10.58346/JISIS.2023.I1.004.
[18] M. Qasim, M. Sajid, An efficient IoT task scheduling algorithm in cloud environment using modified Firefly algorithm, Int. J. Inf. Technol. 1 (2024) 1–10, https://doi.org/10.1007/s41870-024-01758-5.
[19] A.Y. Hamed, M.K. Elnahary, F.S. Alsubaei, H.H. El-Sayed, Optimization task scheduling using cooperation search algorithm for heterogeneous cloud computing systems, Comput. Mater. Contin. 1 (2023) 74, https://doi.org/10.32604/cmc.2022.032215.
[20] Z. Sun, Y. Mei, F. Zhang, H. Huang, C. Gu, M. Zhang, Multi-tree genetic programming hyper-heuristic for dynamic flexible workflow scheduling in multi-clouds, IEEE Trans Serv. Comput. 1 (2024) 1–12, https://doi.org/10.1109/TSC.2024.3394691.
[21] M.A. Abu-Hashem, M. Shehab, M.K.Y. Shambour, M.S.L. Daoud, Abualigah, Improved Black Widow Optimization: an

investigation into enhancing cloud task scheduling efficiency, Sustain. Comput. Inform. Syst. 41 (2024) 100949, https://doi.org/10.1016/j.suscom.2023.100949.
[22] S. Mangalampalli, G.R. Karri, S.N. Mohanty, S. Ali, M.I. Khan, S. Abdullaev, S.A. AlQahtani, Multi-objective Prioritized Task Scheduler using improved Asynchronous advantage actor critic (a3c) algorithm in multi cloud environment, IEEE 1 (2024) 1–12, https://doi.org/10.1109/ACCESS.2024.3355092.
[23] K.G. Krishnasamy, S. Periasamy, K. Periasamy, V. Prasanna Moorthy, G. Thangavel, R. Lamba, S. Muthusamy, A pair-task heuristic for scheduling tasks in heterogeneous multi-cloud environment, Wireless Pers. Commun. 131 (2023) 773–804, https://doi.org/10.1007/s11277-023-10454-9.
[24] C. Yi, T. Zhao, X. Cai, J. Chen, Research on scheduling of two types of tasks in multi-cloud environment based on multi-task optimization algorithm, J. Appl. Anal. Comput. 14 (2024) 436–457, https://doi.org/10.11948/20230266.
[25] Y. Huang, S. Zhang, B. Wang, An improved genetic algorithm with swarm intelligence for security-aware task scheduling in hybrid clouds, Electronics 12 (2023) 2064, https://doi.org/10.3390/electronics12092064.
[26] N. Elsakaan, K. Amroun, A novel multi-level hybrid load balancing and tasks scheduling algorithm for cloud computing environment, J. Supercomput. 1 (2024) 1–41, https://doi.org/10.1007/s11227-024-05990-5.
[27] M. Alam, M. Shahid, S. Mustajab, Security prioritized multiple workflow allocation model under precedence constraints in cloud computing environment, Cluster Comput. 27 (2024) 341–376, https://doi.org/10.1007/s10586-022-03819-5.
[28] H. Gu, M. Zhang, W. Li, Y. Pan, Task offloading and resource allocation based on DL-GA in mobile edge computing, Turk. J. Electr. Eng. Comput. Sci. 31 (2023) 498–515, https://doi.org/10.55730/1300-0632.3998.
[29] P. Pabitha, K. Nivitha, C. Gunavathi, B. Panjavarnam, A chameleon and remora search optimization algorithm for handling task scheduling uncertainty problem in cloud computing, Sust. Comput. Info. Syst. 41 (2024) 100944, https://doi.org/10.1016/j.suscom.2023.100944.
[30] S. Mangalampalli, G.R. Karri, M.V. Ratnamani, S.N. Mohanty, B.A. Jabr, Y.A. Ali, B.S. Abdullaeva, Efficient deep reinforcement learning based task scheduler in multi cloud environment, Sci. Rep. 14 (2024) 21850, https://doi.org/10.1038/s41598-024-72774-5.
[31] R. Ravi, M.J. Pillai, Efficient multiverse electro search optimization for multi-cloud task scheduling and resource allocation, Multimed. Tool. Appl. 1 (2024) 1–26, https://doi.org/10.1007/s11042-024-19901-6.
[32] J. Lwin, Enhancing cloud task scheduling with multi-objective optimization using K-means clustering and dynamic resource allocation, J. Comput. Theo. Appl. 2 (2024) 202–211, https://doi.org/10.62411/jcta.11337.
[33] H. Jia, H. Rao, C. Wen, S. Mirjalili, Crayfish optimization algorithm, Artif. Intell. Rev. 1 (2023) 1919–1979, https://doi.org/10.1007/s10462-023-10567-4.
[34] A.S. Desuky, M.A. Cifci, S. Kausar, S. Hussain, L.M. El Bakrawy, Mud Ring Algorithm: a new meta-heuristic optimization algorithm for solving mathematical and engineering challenges, IEEE Access 10 (2022) 50448–50466, https://doi.org/10.1109/ACCESS.2022.3173401.
[35] S. Mangalampalli, G.R. Karri, U. Kose, Multi objective trust aware task scheduling algorithm in cloud computing using whale optimization, J. King Saud. Univ. Comput. Inf. Sci. 35 (2023) 791–809, https://doi.org/10.1016/j.jksuci.2023.01.016.