Received: January 26, 2024. Revised: March 8, 2024.

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Security and Privacy Considerations in Multimedia Resource Management Using Hybrid Deep Learning Techniques in Cloud Computing

Nallasivan. G [1]* Karpagam. T [2] Geetha. M [3] Sankarasubramanian. R. S [4]
Kannan. R [5] Bhuvanesh. A [6] Poojitha. G [7]

[1]Department of Computer Science and Engineering,
Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Avadi, Chennai, Tamilnadu, India
[2]Department of Artificial Intelligence and Data Science,
R. M. K. College of Engineering and Technology, Thiruvallur, Tamilnadu, India
[3]Department of Electrical and Electronics Engineering,
Sri Eshwar College of Engineering, Coimbatore, Tamil Nadu, India
[4]Department of Mathematics, PSG Institute of Technology and Applied Research, Coimbatore, Tamil Nadu, India
[5]Department of Electrical and Electronics Engineering,
Nehru Institute of Engineering and Technology, Coimbatore, Tamil Nadu, India
[6]Department of Electrical and Electronics Engineering,
PSN College of Engineering and Technology, Tirunelveli, Tamilnadu, India
[7]Department of Artificial Intelligence and Data Science,
R.M.K. College of Engineering and Technology, Tiruvallur, Tamil Nadu, India
* Corresponding author's Email: udhayanallasivan@gmail.com

Abstract: The management of various multimedia assets, such as photos, videos, audio files, and other rich media content, within a cloud computing environment is referred to as managing multimedia resources in the cloud. To suit the needs of applications and users, this entails the effective storage, retrieval, processing, and distribution of multimedia resources. Given the significance of work planning and managing resources in the cloud computing environment, we present a unique hybrid algorithm in this research. Many cloud-based computing systems have made extensive use of traditional scheduling techniques like ant colony optimization (ACO), first come first serve, etc. The cloud gets client tasks at a high rate, so it is important to handle resource allocation for these tasks carefully. Using the improved pelican optimization algorithm, we efficiently distribute the tasks to the virtual machines in this proposed work. The proposed hybrid algorithm (Improved POA + Improved GJO) is then used to distribute and manage the resources (Memory and CPU) as needed by the tasks. According to experimental findings, the accuracy of the proposed technique increases by 1.12%, 2.11%, and 14.2%, respectively. It shows that the proposed method has good accuracy compared with the existing HUNTER, FT-ERM, and RU-VMM approaches.

Keywords: Deep learning algorithms, Resource management, Load balancing, Virtual machines, Task scheduling.

1. Introduction

A large portion of multimedia services are now supplied over the internet thanks to Web 2.0's rapid expansion. The online multimedia systems include a wide range of functions such media content creation, editing, processing, searching, and storage. Supporting such systems has meant placing heavy and diverse demands on processing, storage, and communication capabilities [1-3]. Cloud computing has gained popularity over the last ten years as a viable platform for supplying multimedia services with the resources they need and the quality of service (QoS) they require. Multimedia social programs, online picture and video editing, cloud-based video and picture sharing, and other cloud-based applications are becoming commonplace [4, 5].

With the development of virtual machine (VM) methodology, cloud computing is intended to provide on-demand, QoS assured, and cost-effective software and hardware solutions. The task management issue and the handling of resources issue are two of the main research questions in the area of cloud-based multimedia. On the one hand, managing tasks is accomplished by distributing already-existing resources, seeking additional resources, or releasing surplus resources in response to changes in workload [6].

Cost and QoS are two significant issues in the task management research. On the other side, using the VM allocation strategy solves the problem of resource management. The long-term cost and waiting time are typically the two components of the optimization aim for cloud VM allocation. Despite the fact that cloud-based multimedia platforms have been the subject of several research, the two research concerns we described above have not been taken into account together [7, 8]. Due to the intricacy of the problem, heuristic-based VM allocation algorithms are commonly used for resource management.

Despite the fact that cloud-based multimedia platforms have been the subject of several research, the two research concerns we described above have not been taken into account together [9,10]. Since queue parameters directly relate to QoS and the price of multimedia applications, queuing systems are frequently employed for job management [11]. Due to the intricacy of the problem, heuristic-based VM allocation algorithms are commonly used for resource management. Using IPOA, IGJO, hybrid algorithms, we allocate and manage cloud resources in this research with a primary focus on task scheduling using IPOA. When compared to previous proposed algorithms, our proposed IPOA is more effective in scheduling tasks. To overcome the issues, the proposed hybrid algorithm surpasses peer research methods.

The key contribution of this research are:

- The research introduces a novel hybrid algorithm combining IPOA and IGJO for effective task scheduling and resource management in cloud computing environments. This hybrid approach aims to enhance resource allocation efficiency and task scheduling effectiveness.
- The proposed methodologies focus on dynamic resource allocation and management, considering factors such as CPU and memory utilization. By efficiently distributing tasks among virtual machines and utilizing surplus resources,
- The performance of the proposed hybrid algorithm is evaluated against existing methods such as Round Robin, Throttled, ACO, and Exact Algorithm. The evaluation metrics include average response time, VM utilization, and resource usage efficiency.

The remaining sections of the essay are structured as follows. Task scheduling and resource management-related work is presented at the start of section 2, and the proposed approach and hybrid heuristic methods are covered in section 3. The performance assessment and simulation scenarios are covered in section 4, which is followed by an analysis of the experimental data and a conclusion in section 5.

2. Literature survey

The provision of effective dynamic resource management for infrastructure in software-based networks is [12] one of the most crucial concerns in network virtualization. By combining the Markov-Process with TDMA protocol, the introduced solution (cTMvSDN) improves the handling of resources. The markov-pattern and TDMA slicing system is utilized to forecast upcoming time gaps in order to maximize response time and SDN Quality of service. The disadvantage is as the size of the network grows, maintaining trust relationships between a large number of nodes becomes increasingly challenging.

[13] presented HUNTER, a holistic resource management method for sustainable cloud computing that is based on artificial intelligence (AI). In order to approximate the QoS for a model state and produce the best scheduling options, HUNTER uses a gated graph convolution network. The proposed technique may face challenges in scaling to large-scale cloud environments or adapting to diverse application workloads.

By forcing high-availability in servers and virtual machines, [14] suggested a FT-ERM system that approaches the aforementioned issue from a different angle. A fault-tolerance unit made up of a decision matrix and safe box is proposed. Developing and maintaining a system with proactive failure prediction, dynamic VM migration, and failure tolerance mechanisms can be challenging and costly. overheating.

[15] introduced Sinan, an online, data-driven, QoS-aware cluster manager for interacting cloud microservices. they test Sinan using representative

end-to-end microservice-based applications, on both specialized neighbourhood clusters and massive deployments on Google compute engine (GCE). Implementing a scalable and QoS-aware resource manager like Sinan may introduce additional complexity and overhead to the system.

[16] presented the RU-VMM method, which distributes the loads across the underloaded and overloaded vehicles in order to reduce energy usage. Additionally, RU-VMM makes an effort to prevent pointless VM migrations within the vehicle. One drawback is that the suggested algorithm does not take into consideration the bandwidth and latency involved in virtual machine transfer.

The above literatures show some disadvantages like scalability challenges, complexity, adaptability issues, increased overhead. To overcome the issues, proposed a deep learning-based algorithms to multimedia resource management in cloud. The novelty of our approach lies in the integration of IPOA and IGJO to address these challenges of task scheduling and resource management simultaneously.

3. Proposed methodologies

For task scheduling, we proposed a better IPOA. For resource management and allocation, we also presented the IPOA, IGJO, and hybrid (IPOA+IGJO) methodologies.

3.1 Improved pelican optimization algorithm (IPOA)

A new meta-heuristic optimization method called the POA is motivated by pelican hunting techniques. The world's warm waterways are home to pelicans, which primarily inhabit lakes, rivers, beaches, and marshes. Pelicans typically live in groups and have excellent swimming and flying abilities. They primarily eat fish and have good observational skills and keen eyesight while flying. Once the pelicans have located their prey, they race toward it from a height of 10 to 20 meters before diving headfirst into the ocean to begin their hunt.

(1) ***Initialization:*** If N pelicans in an M-dimensional space, then the location of the i-th pelican is $P_1 = [p_{i1}, p_{i2}, \dots, p_{im}, \dots, p_{iM}]$, and the location P of the N pelicans is written as follows:

$$P = \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_i \\ \vdots \\ P_N \end{bmatrix} = \begin{bmatrix} P_{11}\, P_{12} \dots P_{1m} \dots P_{1M} \\ P_{21}\, P_{22} \dots P_{2m} \dots P_{2M} \\ \vdots \;\; \vdots \;\; \vdots \quad \vdots \quad \vdots \\ P_{i1}\; P_{i2} \dots P_{im} \dots P_{iM} \\ \vdots \;\; \vdots \;\; \vdots \quad \vdots \quad \vdots \\ P_{N1} P_{N2} \dots P_{Nm} \dots P_{NM} \end{bmatrix}, i = 1,2, \dots \dots, N \tag{1}$$

The location updates of the pelican are denoted as

$$P_{im} = low_m + random \cdot (up_m - low_m), \quad i = 1,2, \dots, M; \tag{2}$$

Where random is a value chosen at random between (0,1) and low_m and up_m are the pelican's search ranges.

(2) ***Moving towards prey:*** The pelican rushes at the prey from a great height after locating it during this phase. It is defined as

$$P_{im}^{t+1} = \begin{cases} P_{im}^t + rand \cdot \left(S_m^t - \lambda \cdot P_{im}^t\right), & F(P_s) < f(P_i) \\ P_{im}^t + rand \cdot \left(P_{im}^t - \lambda \cdot S_m^t\right), & F(P_s) \geq f(P_i) \end{cases} \tag{3}$$

(3) ***Winging on the water surface:*** The fish are lifted up by the pelicans till they reach the top of the water, and they then sweep it up in their throats pouch. The pelicans' hunting habits is simulated mathematically.

$$P_{im}^{t+1} = p_{im}^t + \gamma \cdot \left(\frac{T-t}{T}\right) \cdot (2 \cdot random - 1) \cdot P_{im}^t \tag{4}$$

The efficiency of the optimization process could be further increased after making changes to the original POA algorithm. The following is the precise improvement strategy.

(1) ***Initialization strategy***: After the Tent chaotic mapping is implemented, the randomly created initialization technique in the basic POA is replaced with the Tent chaotic map, and Eq. (6) can be recast as follows:

$$p_{im} = low_m + Tent \cdot (up_m - low_m), \quad i = 1,2, \dots, N; \;\; m = 1,2, \dots, M; \tag{5}$$

$$Tent^{t+1} = \begin{cases} \frac{Tent^t}{2}, & Tent^t \in [0, z] \\ \frac{(1-Tent^t)}{(1-z)}, & Tent^t \in [z, 1] \end{cases} \tag{6}$$

The POA algorithm's global search performance is now enhanced by initializing the pelicans' positions utilizing the Tent chaotic map.

(2) ***Moving towards prey:*** At this point, the pelican may continuously update its position thanks to the dynamic weight factor θ. θ lowers adaptively toward the conclusion of the iteration. It is possible to rewrite Eq. (7) as follows:

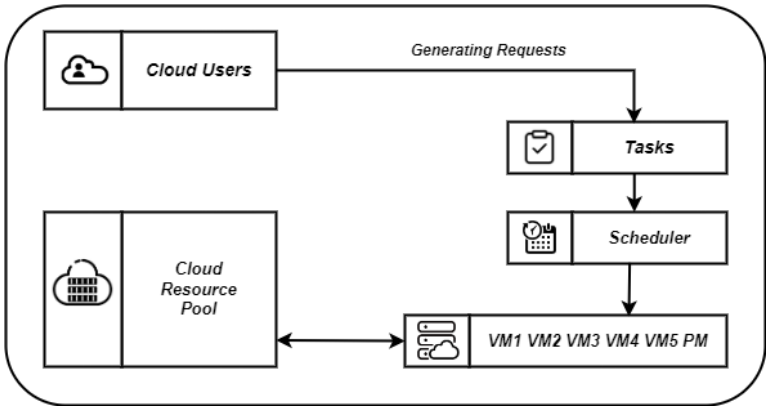


Figure. 1 Architecture of proposed method

$$P_{im}^{t+1} = \begin{cases} \theta = \frac{e^{2(1-t/T)} - e^{-2(1-t/T)}}{e^{2(1-t/T)} + e^{-2(1-t/T)}} \\ P_{im}^{t} + rand \cdot \left(S_m^t - P_{im}^t\right) \cdot \theta, \ \ f(P_s) < f(P_i) \\ P_{im}^{t} + rand \cdot \left(P_{im}^t - S_m^t\right) \cdot \theta, \ \ f(P_s) \geq f(P_i) \end{cases} \quad (7)$$

A Fig. 1 demonstrates the architecture of proposed methodology.

3.2 Improved golden jackal optimization (IGJO)

GJO mimics biological swarm intelligence, which is based on the golden jackal's hunting style. The hunt involves three stages: locating the prey, besieging and arousing it, and attacking it. The GJO algorithm's mathematical framework is developed in the sections that follow.

3.2.1. Search model

The prey's random position throughout the initial phase.

3.2.2. Exploration stage

It is difficult to catch the prey since jackals naturally have the capacity to pursue their prey. The jackals will therefore be waiting to capture some other victim. The following equations (|E|>1) can be used to define the hunting behaviour:

$$Y_1(t) = Y_M(t) - E \cdot |Y_M(t) - rl \cdot Prey(t)| \quad (8)$$

$$Y_2(t) = Y_{FM}(t) - E \cdot |Y_{FM}(t) - rl \cdot Prey(t)| \quad (9)$$

$Y_M(t)$ and $Y_{FM}(t)$ stand for the positions of the female and male jackals, respectively, while t stands for the algorithm's current iteration. $Prey(t)$ indicates the hunting location vector, and $Y_1(t)$ and $Y_2(t)$ are, of course, updated jackal positions.

It will be determined how much prey has in escape energy (E) by:

$$E = E_1 \cdot E_0, \ \ E_0 = 2 \cdot r - 1 \quad (10)$$

$$E_1 = c_1 \cdot \left(1 - \frac{t}{T}\right) \quad (11)$$

E_1 denotes the decrease in the prey's energy, E_0 denotes a random integer between -1 and 1, T denotes the greatest amount of iterations, c_1 denotes a fixed amount with a value of 1.5.

$$rl = \frac{5 \cdot LF(y)}{100} \quad (12)$$

$$LF(y) = \frac{\mu \cdot \sigma}{100. \left| v^{\left(\frac{1}{\beta}\right)} \right|} \quad (13)$$

$$\sigma = \left\{ \frac{\Gamma(1+\beta) \cdot \sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \cdot \beta \cdot \left(2^{\beta-1}\right)} \right\}^{\frac{1}{\beta}} \quad (14)$$

$$Y(t+1) = \frac{Y_1(t) + Y_2(t)}{2} \quad (15)$$

The new position of the prey with relation to the jackals is represented by $Y(t+1)$.

3.2.3. Exploitation

Golden jackals' pestering of prey reduces their ability to flee. Here (|E|1) is a model for how jackals behave when pursuing and consuming their prey:

$$Y_1(t) = Y_M(t) - E \cdot |rl \cdot Y_M(t) - Prey(t)| \quad (16)$$

$$Y_2(t) = Y_{FM}(t) - E \cdot |rl \cdot Y_{FM}(t) - Prey(t)| \quad (17)$$

3.2.4. Transition from exploration stage to convergence and exploitation

The exploration phase is switched over to the GJO algorithm's usage of the prey's volatile energy. The prey loses a great deal of energy as it flees. In each repetition, the beginning energy E0 is arbitrarily

deviated between 1 and 1. With more repetitions, prey loses energy trying to escape. The system then uses a pair of jackals to guess where the prey is. The response from each candidate alters its place in relation to the jackal pair. The exploratory and extraction stages are supplied by lowering E1 from 1.5 to 0. In conditions E>1 and E1, the pair of jackals veer away from the prey, respectively. The GJO algorithm terminates when the convergence requirements are satisfied.

3.2.5. Overview of improved GJO (IGJO)

The imbalance between exploitation and exploration, as well as the standard GJO method's tendency to become stuck in premature convergence, are concerns. In this paper, Rosenbrock's direct rotational (RDR) approach is employed to enhance the effectiveness of the conventional GJO algorithm against these issues. The current phase has been completed, and the identification foundation is now being examined to determine the overall impact of successful stages on every of the dimensions. The following changes have been made to the orthonormal basis:

$$x^{k+1} - x^{k+} = \sum_{i=1}^{n} \lambda_i \cdot d_i \tag{18}$$

The directions are given in the equation below. $x^{k+1} - x^{k+}$ denotes the point with the most advantageous search direction, whereas λ_i denotes the quantity of successful parameters. It is positioned in the correct search direction as a result.

$$pi = \begin{cases} d_i, & \lambda_i = 0 \\ \sum_{j=0}^{n} \lambda_j \cdot d_j & \lambda_i \neq 0 \end{cases} \tag{19}$$

The next step is to update the search outcomes using the Gram-Schmidt normalization technique.

$$qi = \begin{cases} p_i, & i = 1 \\ p_i = \sum_{j=1}^{i-1} \frac{q_j^T \cdot p_i}{q_j^T \cdot q_i} & i \geq 2 \end{cases} \tag{20}$$

Following is a definition of the modified and normalized search criteria.

$$d_i = \frac{q_i}{\|q_i\|},\ i = 1,2,3,\dots,n. \tag{21}$$

This method updates the local search and then runs the search operation until the algorithm's convergence criterion is satisfied in the new opposite direction.

3.3 Proposed model

In this proposed approach, jobs are planned on the virtual machines, and the proposed algorithms also effectively use resources like CPU and memory. For the efficient operation of the proposed approaches, various kind of virtual machines on a PM that can interact with the scheduler were used in this case. The tasks arrive in batches of ten, and the period between each batch arrival is constant. Prior to allocating the needed resources, tasks are first efficiently planned on virtual machines. Fig. 1 shows the proposed research. The incoming task requires n cloud resources. The IGJO method uses the excess res3[] mapping while the IPOA algorithm uses the excess res2 and excess res1 mappings. On the other hand, the hybrid method makes advantage of each of the three resources mappings discussed previously.

3.3.1. IPOA algorithm for scheduling of VMs

Here, we utilize the IPOA method to schedule incoming tasks on VMs swiftly. Efficiently assigning duties received by the cloud is challenging due to numerous incoming tasks needing distribution across VMs (fvm0, vm1, vm2,..., vmkg). Our proposed IPOA method ensures balanced scheduling, crucial for effective workload distribution. Testing was conducted on a private cloud handling batches of ten jobs, adaptable to larger volumes. The number of VMs used impacts clustering.

Each iteration, every cluster will determine the smallest of these VMs, known as the global best (GB), and the least loaded VM, known as the local best (LBz). The VM connected to GB receives the subsequent task. Until all of the jobs are completed, the same procedure is repeated. This algorithm's temporal complexity is 0(n.z). The IPOA algorithm's time complexity will be 0(n) in polynomial time because z is a constant.

3.3.2. IPOA algorithm for resource management and allocation

In the cloud environment, tasks demand dynamic resources at a quick rate, and meeting these demands is a difficult issue. To complete tasks for customers, VMs need sufficient resources, which are handled by the proposed IPOA. Consider that there is a res pool that serves as both a resource repository and a supplier of the resources needed by the jobs. For the purpose of carrying out a work, every of the VMs has a minimum of 2 resources (CPU and Memory). Each VM won't consume all the resources when completing activities, therefore the extra ones that are left over can be put to use for upcoming tasks.

Let's create clusters fc1, c2, c3,..., czg from VMs fvm0, vm1, vm2,..., vmkg that operate in both parallel and sequential modes for improved efficiency. We require at least 2 clusters for execution in parallel mode. The job is instantly started by VMs if the next resource requirements match the unused resources, else, the resources are pulled from a pool of resources for the execution.

As a result, communication between the virtual machine and the cloud resource pool is greatly reduced as we dynamically use the resources. The global and local best frequently provide maximum and minimum values in IPOA, and it also depends on the application. An increase in the amount of communication between virtual machines and the cloud resource pool.

3.3.3. IGJO Algorithm for resource management and allocation

Jackals always maintain a calm demeanour while moving slowly, and this is known as the seeking mode in jackal behaviour. Jackals hunt prey at great speeds when they detect its presence (a resource match occurs), and this activity is reflected in the tracing mode. While the searching mode watches for a chance to catch a victim, the tracing mode functions similarly to the IGJO algorithm. Instead of tracing mode, we primarily focus on searching mode in the proposed IGJO algorithm.

IGJO's seeking mode explores four memory types. The searching memory pool (SMP) holds additional VM resources not in excess res1 or res2. Status is saved in the seeking range of the chosen dimension (SRD), triggering task execution when updated (CDC condition). The position of the golden jackal is recorded in the SPC, updated with each seeking mode update. Tracing mode processes seeking mode results. Remaining resources may meet future demands, reducing resource borrowing time significantly.

The IGJO algorithm thereby overcomes the IPOA algorithm's aforementioned drawbacks. IGJO will enhance the dynamic allocation and handling of cloud resources because its seeking mode match ratio is higher than that of the IPOA's top two matching rules, excess res1 and excess res2.

3.3.4. HYBRID (IPOA+IGJO) algorithm for resource allocation and management

The proposed hybrid algorithm, which combines the benefits of both IPOA and IGJO approaches, can overcome the limitations of the IPOA and IGJO algorithms. Therefore, the hybrid technique offers improved resource allocation efficiency with shorter overall execution times. The hybrid method, combining IPOA and IGJO, reduces communication overhead between the resource pool and VMs by only comparing exact matches between excess resources 1, 2, and future resource demands. However, in worst-case scenarios, it may degrade resource allocation performance, causing significant system delays. To address this, we consider excess res3[], containing additional resources. Consequently, compared to IPOA and IGJO individually, the hybrid method shows superior performance. To execute the lines 18–25 and 26–29 concurrently utilizing Python threads (using both IPOA and IGJO), we must simultaneously apply both techniques. As a result, when IPOA and IGJO methods are compared independently, excess res3[] has a greater chance of matching future resource demands.

Table 1. PySim configuration information

Machine	GHz	RAM(GB)	Stroage (GB)
Load Balancer	3.40	8	100
Client Machine	2.80	4	100
Server 2	3.40	16	200
Server 1	3.40	16	200

Table 2. VMs configuration information

VM Type	MIPS	RAM(GB)	Stroage(GB)
Medium	1000	1	30
Small	500	0.5	20
X. Large	2000	3	50
Large 1	1500	2	40
Extra Large	2500	4	50

4. Performance evaluation

On a simulator that provides a real-time cloud environment scenario, the proposed work is tested. Python Simulator, or PySim for short, is the term given to the entire simulation scenario that has been written in Python. Four actual machines make up the experimental setup, which is connected to them in a variety of ways. The configuration information's of the experiment's setup are provided in Table 1. Here, the term "task" refers to carrying out a task that needs a number of resources. We regarded CPU and Memory as two resource categories that are required for carrying out a task. A large pool of resources in the cloud is used to provide resources.

Ten tasks were taken into account, and they randomly demanded cloud resources. Based on the proposed techniques, a load balancer manages the allocation and administration of these resources.

Table 3. Workload setup for scheduling

User Base	Region	Online Users during peak hours	Online Users during non-peak hours
North America	0	135000	13500
Europe	2	255000	25500
South America	1	125000	12500
Africa	4	30000	3000
Asia	3	535000	53500
Oceania	5	10000	1000

Table 4. VMs usage

Sl. No.	Throttled	Round Robin	ACO	Exact Algorithm	Proposed Algorithm
VM1	1182	254	356	253	254
VM2	76	254	289	254	254
VM3	8	253	389	254	253
VM4	2	253	180	254	254
VM5	0	254	54	253	253

Table 5. Comparison of proposed and other algorithms for average response time

Algorithms	Average Response Time
Round Robin	364.85ms
Throttled	365.52ms
Exact Algorithm	365.87ms
ACO	362.67ms
Proposed Algorithm	360.11ms

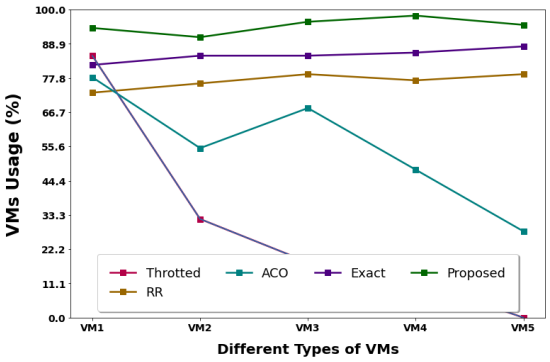


Figure 2. Average utilization of VMs

In this regard, we utilized five distinct kinds of virtual machines, and Table 2 displays their configuration information. Table 3 displays the workload configuration utilized to carry out the scheduling experiment.

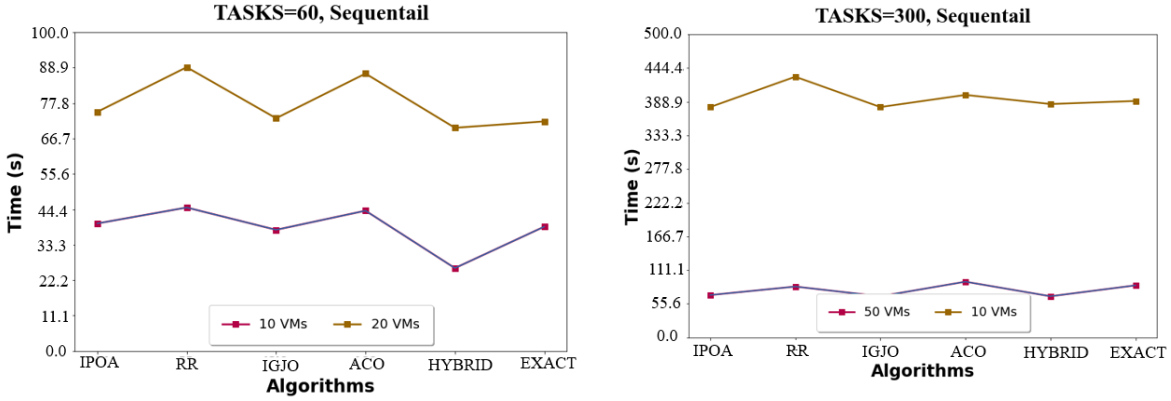
Table 4 indicates that Round Robin and proposed IPOA methods outperform the Throttled approach in task scheduling efficiency. While Round Robin and IPOA yield similar outputs, Round Robin's lack of VM status checking leads to server queuing. IPOA, by considering VM status and cluster-based comparisons, prevents queuing and improves task assignment. ACO assigns tasks based on lower pheromone contents in VMs, akin to MPSO, albeit with higher response times.

The Table 5 shows the Average response duration for that configuration. The proposed IPOA method not only effectively distributes the load among the virtual computers, as can be seen from Tables 4 and 5, but it also achieves a superior average response time.

The experiment is run again with various sets of virtual machines and jobs, and the outcomes are reliable. The usage of VMs with various combinations of 1000, 2000, and 3000 workloads was then examined. The research's employment of several algorithms on VMs is depicted in Fig. 2. As can be seen from Fig. 2, when compared to other algorithms, the Throttled and ACO methods do not consistently use the VMs.

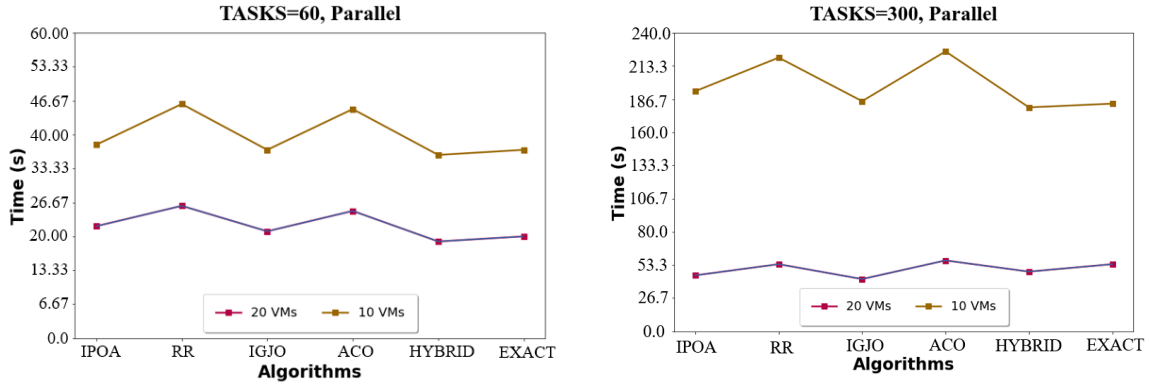
While RR and Exact Methods effectively utilize VMs, our IPOA method is even more efficient with VMs and ATR. IGJO's Seeking mode takes longer than Tracing mode, thus not considered for IPOA scheduling. Similarly, the proposed hybrid, combining IPOA and IGJO, is also not accounted for in scheduling. Experiments in sequential and parallel modes determine average response time, displayed in Figs. 3 and 4.

We employed 10 and 20 virtual machines in two clusters with 60 incoming jobs in Figure 3a. With 300 incoming jobs, we employed 10 and 50 VMs across two clusters in Figure 3b. The two best VMs from every cluster are selected in IPOA's best match (GB) case technique so that the resources can match the anticipated demands. We compare resource demands and excess in each cluster, using the IGJO strategy for remaining VMs. Specifically, we compare each cluster's extra resources to forthcoming demands. The experiment runs parallel with threads allocated asymmetrically based on cluster count, crucial for efficient resource management and speed enhancement per cluster.



(a) (b)

Figure. 3 Squentail analysis: (a) Sequentail Analysis with 60 tasks and Squentail analysis with 300 tasks



(a) (b)

Figure. 4 Parallel analysis: (a) Parallel analysis with 60 tasks and Parallel analysis with 300 tasks

As can be seen from Fig. 4, parallel mode execution will result in a faster average reaction time than sequential mode. Figs. 3 and 4 show that, when compared to the branch-and-bound based Exact approach, the IPOA, IGJO, and hybrid (IPOA+IGJO) algorithms provide workable solutions in terms of average reaction time. Additionally, our hybrid approach is faster than all other algorithms.

Accordingly, the resource usage factor is examined in relation to proposed methods. The resource use with 1000, 2000, and 3000 jobs is shown in Figs. 5 (a), (b), and (c), respectively. In each of the three scenarios, RR uses the maximum amount of cloud resource pool resources. ACO uses approximately an equal number of resources from the unused VM resources as well as the cloud resource pool.

IGJO surpasses IPOA in resource utilization by effectively employing inactive cluster resources. The hybrid method outperforms traditional approaches, as demonstrated by the benchmark Exact method in Fig. 5, showcasing superior resource efficiency compared to previous methods considered for evaluation.

4.1 Time complexity analysis

In the proposed hybrid method, IPOA and IGJO approaches are the foundation. The proposed hybrid method's time complexity is therefore O (amount of tasks) *O (amount of clusters) *O (amount of VMs) *O (amount of'm'iterations) *O(IPOA)*O(IGJO), which is equal to O(n)*O(Cz)*O(VMk)*O('m') *O (n, Cz) *(n, Cz). The formula becomes O(n*Cz*VMk*m).

The computational time for the three averages, best, and worst methods out of all those proposed is shown in Fig. 6. Utilizing the proposed and cutting-edge benchmark techniques, we tested various job combinations and VM counts. In Fig. 6, worst-case execution time occurs when surplus resources don't match anticipated needs. RR's constant execution time stems from its reliance on cloud resources. ACO's time consistency results from variable pheromone content. Exact algorithm's uniform execution time is due to exhaustive resource allocation attempts.

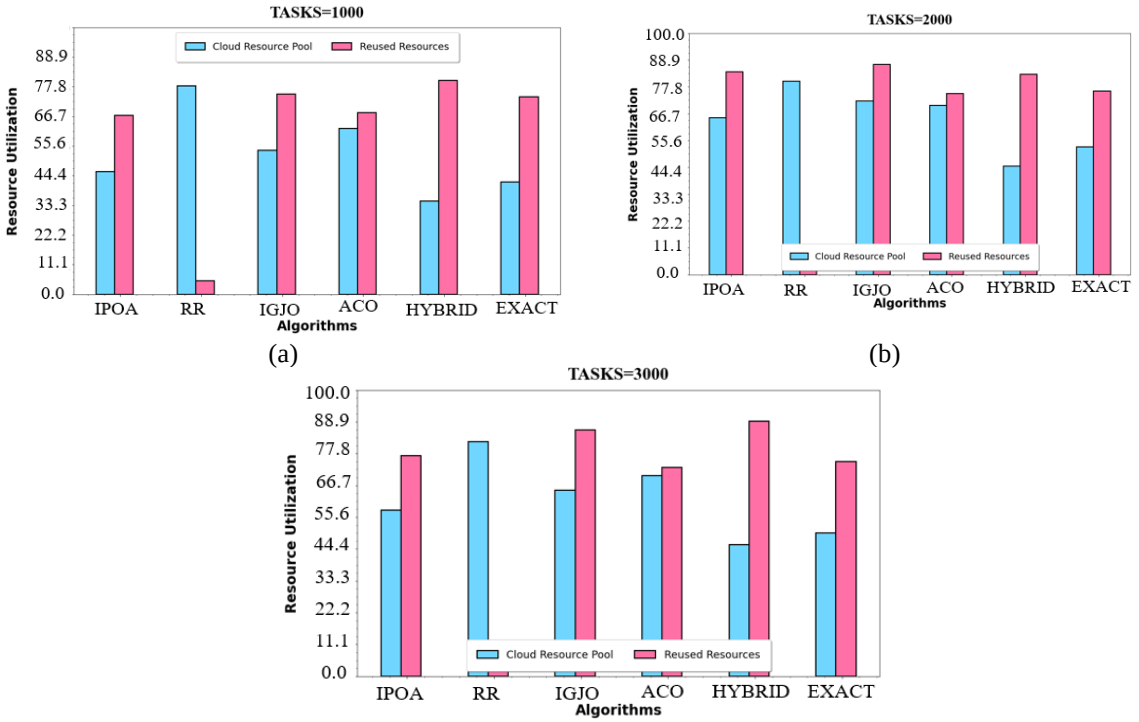


Figure. 5 Average resource usage: (a) Resource usage with 1000 tasks, (b) Resource usage with 2000 tasks, and (c) Resource usage with 3000 tasks

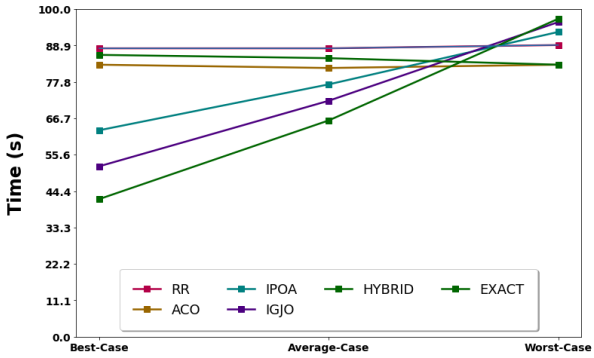


Figure. 6 Execution time analysis of proposed algorithms

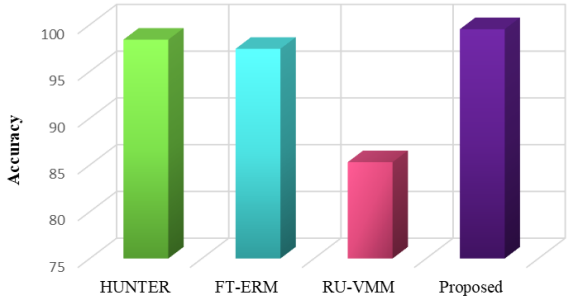


Figure. 7 Comparison in terms of accuracy

4.2 Comparative analysis

The efficiency of the suggested model is compared to that of HUNTER (13), FT-ERM (14), and RU-VMM (16) in terms of accuracy, and scalability.

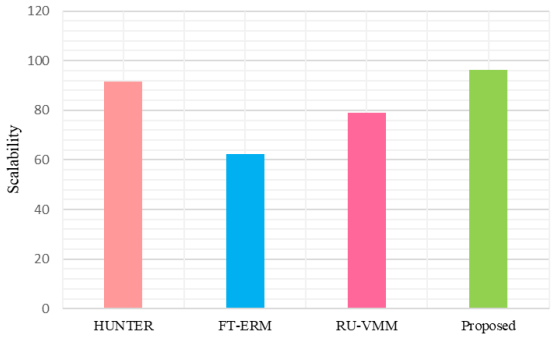


Figure. 8 Comparison in terms of scalability

Fig. 7 shows the accuracy comparison of the proposed technique with the existing techniques, including HUNTER, FT-ERM, and RU-VMM. The accuracy of the proposed technique increases by 1.12%, 2.11%, and 14.2%, respectively. It shows that the proposed method has good accuracy compared with the existing HUNTER, FT-ERM, and RU-VMM approaches.

Fig. 8 compares scalability between existing HUNTER, FT-ERM, and RU-VMM proposed method. The proposed method exhibits superior scalability, efficiently accommodating growing resource requirements without compromising performance or stability. The accuracy of the proposed technique increases by 4.78%, 35.1%, and 17.98%, compared to existing HUNTER, FT-ERM, and RU-VMM techniques respectively.

5. Conclusion

In this paper, a novel hybrid algorithm that combines improved pelican optimization algorithm (IPOA) and improved golden jackal optimization (IGJO), to enhance efficiency in resource allocation. The proposed methodologies focus on dynamic resource allocation, considering factors such as CPU and memory utilization, and aim to improve resource allocation efficiency and task scheduling effectiveness. The experiments conducted demonstrate the effectiveness of the proposed approach compared to existing algorithms such as Round Robin, Throttled, ACO, and exact algorithm. Results indicate that the proposed hybrid algorithm outperforms these methods in terms of average response time, VM utilization, and resource usage efficiency. Additionally, the proposed algorithm shows improved accuracy and scalability compared to existing techniques like HUNTER, FT-ERM, and RU-VMM. According to experimental findings, the accuracy of the proposed technique is higher than that of the existing HUNTER, FT-ERM, and RU-VMM approaches by 1.12%, 2.11%, and 14.2%, respectively. Future research will focus on integrating cutting-edge technologies like blockchain and edge computing to improve multimedia resource management in cloud computing environments' security and privacy.

Conflicts 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.

Author contributions

The following statements should be used as follows: “Conceptualization, G. Nallasivan and T. Karpagam; methodology, M. Geetha; software, R. S. Sankarasubramanian; validation, R. Kannan, A. Bhuvanesh, and G. Poojitha; formal analysis, G. Nallasivan; investigation, T. Karpagam; resources, M. Geetha; data curation, R. S. Sankarasubramanian; writing—original draft preparation, R. Kannan; writing—review and editing, A. Bhuvanesh; visualization, G. Poojitha; supervision, G. Nallasivan; project administration, T. Karpagam; funding acquisition, M. Geetha”, etc.

Acknowledgments

The author would like to express his heartfelt gratitude to the supervisor for his guidance and unwavering support during this research for his guidance and support.

References

[1] A. Montazerolghaem, M. H. Yaghmaee, and A. Leon-Garcia, “Green cloud multimedia networking: NFV/SDN based energy-efficient resource allocation”, *IEEE Transactions on Green Communications and Networking*, Vol. 4, No. 3, pp. 873-889, 2020.

[2] Y. Zhang, W. Hua, Z. Zhou, G. E. Suh, and C. Delimitrou, “Sinan: ML-based and QoS-aware resource management for cloud microservices”, In: *Proc. of the 26th Acm International Conference on Architectural Support For Programming Languages and Operating Systems*, pp. 167-181, 2021.

[3] R. T. Rodoshi, T. Kim, and W. Choi, “Resource management in cloud radio access network: Conventional and new approaches”, *Sensors*, Vol. 20, No. 9, p. 2708, 2020.

[4] L. Ding, Z. Wang, X. Wang, and D. Wu, “Security information transmission algorithms for IoT based on cloud computing”, *Computer Communications*, Vol. 155, pp. 32-39, 2020.

[5] A. Mijuskovic, A. Chiumento, R. Bemthuis, A. Aldea, and P. Havinga, “Resource management techniques for cloud/fog and edge computing: An evaluation framework and classification”, *Sensors*, Vol. 21, No. 5, p. 1832, 2021.

[6] N. Gholipour, E. Arianyan, and R. Buyya, “A novel energy-aware resource management technique using joint VM and container consolidation approach for green computing in cloud data centers”, *Simulation Modelling Practice and Theory*, Vol. 104, p. 102127, 2020.

[7] X. Xu, Y. Chen, Y. Yuan, T. Huang, X. Zhang, and L. Qi, “Blockchain-based cloudlet management for multimedia workflow in mobile cloud computing”, *Multimedia Tools and Applications*, Vol. 79, pp. 9819-9844, 2020.

[8] P. J. Maenhaut, B. Volckaert, V. Ongenae, and F. D. Turck, “Resource management in a containerized cloud: Status and challenges”, *Journal of Network and Systems Management*, Vol. 28, pp. 197-246, 2020.

[9] G. Rjoub, J. Bentahar, O. A. Wahab, and A. S. Bataineh, “Deep and reinforcement learning for automated task scheduling in large-scale cloud computing systems”, *Concurrency and Computation: Practice and Experience*, Vol. 33, No. 23, p. e5919, 2021.

[10] H. Mia and F. Faisal, “Digital Human Resource Management: Prospects & Challenges for

Garments Industries in Bangladesh", *European Journal of Business and Management*, Vol. 12, No. 7, pp. 18-25, 2020.

[11] Q. Zhang, L. Gui, F. Hou, J. Chen, S. Zhu, and F. Tian, "Dynamic task offloading and resource allocation for mobile-edge computing in dense cloud RAN", *IEEE Internet of Things Journal*, Vol. 7, No. 4, pp. 3282-3299, 2020.

[12] F. Faraji, A. Javadpour, A. K. Sangaiah, and H. Zavieh, "A solution for resource allocation through complex systems in fog computing for the internet of things", *Computing*, pp. 1-25, 2023.

[13] S. Tuli, S. S. Gill, M. Xu, P. Garraghan, R. Bahsoon, S. Dustdar, ... and N. R. Jennings, "HUNTER: AI based holistic resource management for sustainable cloud computing", *Journal of Systems and Software*, Vol. 184, p. 111124, 2022.

[14] D. Saxena, I. Gupta, A. K. Singh, and C. N. Lee, "A fault tolerant elastic resource management framework toward high availability of cloud services", *IEEE Transactions on Network and Service Management*, Vol. 19, No. 3, pp. 3048-3061, 2022.

[15] Y. Zhang, W. Hua, Z. Zhou, G. E. Suh, and C. Delimitrou, "Sinan: ML-based and QoS-aware resource management for cloud microservices", In: *Proc. of the 26th ACM International Conference on Architectural Support For Programming Languages and Operating Systems*, pp. 167-181, 2021.

[16] S. K. Pande, S. K. Panda, S. Das, K. S. Sahoo, A. K. Luhach, N. Z. Jhanjhi, ... and S. Sivanesan, "A Resource Management Algorithm for Virtual Machine Migration in Vehicular Cloud Computing", *Computers, Materials & Continua*, Vol. 67, No. 2, 2021.