IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 17 April 2024, accepted 27 April 2024, date of publication 6 May 2024, date of current version 17 May 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3397696

RESEARCH ARTICLE

Optimizing Task Scheduling and Resource Utilization in Cloud Environment: A Novel Approach Combining Pattern Search With Artificial Rabbit Optimization

SANTOSH KUMAR PAUL[1], SUNIL KUMAR DHAL[1], SANTOSH KUMAR MAJHI[2], (Senior Member, IEEE), ABHIJEET MAHAPATRA[3], PRADOSH KUMAR GANTAYAT[4], AND SWARUPA PANDA[5]

[1]Faculty of Science, Sri Sri University, Cuttack, Odisha 754006, India
[2]Department of Computer Science and Information Technology, Guru Ghasidas Vishwavidyalaya (Central University), Bilaspur, Chhattisgarh 495009, India
[3]Department of Computer Science and Engineering, Veer Surendra Sai University of Technology, Burla, Odisha 768018, India
[4]Department of Artificial Intelligence and Data Science, Faculty of Science & Technology (IcfaiTech), The ICFAI Foundation for Higher Education, Hyderabad, Telangana 501203, India
[5]Department of Computer Science and Engineering, Government College of Engineering, Keonjhar, Odisha 758002, India

Corresponding authors: Santosh Kumar Majhi (smajhi_cse@ieee.org) and Pradosh Kumar Gantayat (drpradosh@ifheindia.org)

ABSTRACT The increasing demand for Cloud service with sudden resource requirements of Virtual Machines (VMs) with different resource types and sizes may create an unbalanced state in the Cloud datacenters. In turn, it will lead to low resource utilization and slow down the server's performance. This research article proposes an enhanced version of the Artificial Rabbit Optimization (ARO) called Improved Artificial Rabbit Optimization based on Pattern Search (IARO-PS), where ARO has been utilized to schedule the dynamically independent requests (tasks) for overcoming the challenges discussed above and a Pattern Search (PS) method has been hybridized to address the shortcomings of ARO and to provide better exploration-exploitation balance. The initial step in the proposed approach is to employ a load balancing strategy by dividing the workloads (user requests) across the available VMs. The next step utilizes the IARO-PS method to map the workloads (user requests) onto the optimal VMs for the scheduling process to carry out across the diverse resources. A standard benchmark function (CEC2017) is used to assess the IARO-PS technique's efficacy. A comprehensive evaluation has been carried out by taking an available real-world dataset having different specifications of tasks in the CloudSim to evaluate the performance of the methodology. Additionally, a simulation-based comparison is carried out with various metaheuristic-based workload scheduling methods like Genetic Algorithm (GA), Bird Swarm Optimization (BSO), Modified Particle Swarm Optimization based on Q-learning (QMPSO), and Multi-Objectives Grey Wolf Optimizer (MGWO). Based on the simulations, the IARO-PS algorithm performed better than the previously mentioned algorithms, reducing makespan by 10.45% (GA), 2.31% (QMPSO), 4.35% (MGWO), 15.35% (BSO), and 4.17% (GA), 1.03% (QMPSO), 1.44% (MGWO), 7.33% (BSO), in both homogeneous and heterogeneous surroundings, respectively, and improving resource utilization by 36.74% (GA), 14.31% (QMPSO), 19.75% (MGWO), 45.23% (BSO) and 12.17% (GA), 6.02% (QMPSO), 9.10% (MGWO), 19.39% (BSO). Furthermore, statistical evaluation through Friedman's test and Holm's test has also been carried out showcasing the decrease in makespan and an increase in VM utilization, which are the outcomes of the simulated experimental study.

The associate editor coordinating the review of this manuscript and approving it for publication was Zhengmao Li.

INDEX TERMS Artificial rabbits optimization, cloud computing, load balancing, pattern search, resource management, task scheduling.

I. INTRODUCTION

Cloud computing is considered by many researchers and corporate organizations to be one of the most innovative computer concepts. The pay-per-use model, dynamic resource scalability, wide network connectivity, and availability of numerous resources contributed to the widespread adoption of cloud computing [1]. Cloud users can access all of these services on a reasonable pay-per-use basis. In this model, a contract between cloud users and the broker (service provider) develops a Service Level Agreement (SLA). This agreement specifies the Quality-of-Service (QoS) requirements that must be attained and establishes penalties for noncompliance. It is enhanced as an abstract entity to provide varying degrees of services to all customer domains [2]. A cloud makes use of datacenter equipment and software to provide services and applications that are made available online via the foundational hardware and software [3]. Three main service models are used to provide the hardware or software applications. Customers can get ready-to-use apps through Software-as-a-Service (SaaS). In addition, Platform-as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS) are used to supply the system software and hardware, respectively [4]. It is termed Everything-as-a-Service (XaaS) when these services are made available to different walks of life as a utility. Various deployment models are used to supply services depending on the demands and specifications of the customer. A public cloud makes its cloud services available to the general public under a certain pricing plan. An organization is the only one to use a private cloud. In contrast, a hybrid cloud is sometimes referred to as the best of both cloud architectures.

Users are becoming more and more numerous every day as the necessity for computing grows. Ensuring a balance between workloads is of utmost importance, as it involves addressing the growing requirements of the users. Cloud datacenters should offer effective on-demand assignments safely and flexibly to counteract clients' varying demands [5]. With the aid of virtualization technologies, it might be achievable. The current infrastructure's capability to carry out duties is increased by the virtualization technique. Without requiring the underlying hardware, it allows users to configure, access, and build an abstract surrounding for the VMs [6]. However, using several real hosts, cloud datacentres may support varying VMs with different characteristics. Because of the different possible specifications and unpredictable VM usage, servers in datacenters can be in an unbalanced situation when it comes to resource usage. Low resource utilization and performance deterioration might result from unbalanced server conditions [7]. These issues can be resolved by implementing a load balancing method. This method will evenly distribute the workload among VMs and keep server loads consistent.

Load balancing is the process that distributes local workloads evenly among cloud VMs. By effectively using the resources, it ensures that not a single node is either overloaded, underloaded, or idle [8]. To meet load-balancing goals, an efficient scheduling mechanism should be integrated with the load-balancing algorithm. As a result, it can evenly distribute duties among all of the VMs by making the most use of the resources at hand. In addition to balancing the loads, a suitable load balancing method improves VM utilization and reduces makespan and reaction time. As such, it improves the system's overall performance. Table 1 presents the list of all the acronyms used in this research article along with their respective descriptions.

A. MOTIVATION

Due to the complexity of the QoS parameters involved, scheduling is considered an NP-hard issue [9], [10]. As such, no particular algorithm provides an optimal solution. It is thought that the hybrid metaheuristic and metaheuristic algorithms perform better on their target problem [11]. This work suggests using the PS algorithm in conjunction with an enhanced version of ARO. The PS was selected due to its resilience to local optima and its ability to explore. Because it has a lower computational time complexity than IARO, the PS is selected to overcome this shortcoming. By balancing the local and global optima, the suggested hybrid strategy enhances exploration capability at the expense of exploitation. The fitness function chosen for the particular situation determines how resilient the implemented IARO-PS is. The authors employ a fitness function considering VM utilization and makespan, which are inversely related to each other, for the task scheduling strategy. The authors compare the effectiveness of the present hybrid strategy with other load-balancing and scheduling methods already in use, including BSO, GA, MGWO, and QMPSO.

B. CONTRIBUTIONS

To summarize, the salient features of this paper are outlined as follows:

- This paper proposes an enhanced version of the ARO using a PS algorithm, called IARO-PS to schedule the dynamically independent requests (tasks) for optimizing task scheduling and load balancing parameters in Cloud environment,
- The IARO-PS has been verified using standard benchmark function CEC2017 for its effectiveness,
- The proposed IARO-PS based task scheduling and load balancing method takes a real-time dataset (NASA) [43] to map the workloads onto appropriate VMs, and evenly distributes the workload across resources by taking into account the job compatibility of both underloaded and overloaded VMs,

TABLE 1. List of acronyms used.

Sl. No.	Acronyms	Descriptions
1.	VM	Virtual Machine
2.	ARO	Artificial Rabbit Optimization
3.	PS	Pattern Search
4.	IARO-PS	Improved Artificial Rabbit Optimization using Pattern Search (*proposed*)
5.	GA	Genetic Algorithm
6.	BSO	Bird Swarm Optimization
7.	QMPSO	Modified Particle Swarm Optimization based on Q-learning
8.	SLA	Service Level Agreement
9.	QoS	Quality-of-Service
10.	SaaS	Software-as-a-Service
11.	PaaS	Platform-as-a-Service
12.	IaaS	Infrastructure-as-a-Service
13.	XaaS	Everything-as-a-Service
14.	SA	Simulation Annealing
15.	SG-PBFS	Shortest Gap-Priority Based Fair Scheduling
16.	PBFS	Priority-Based Fair Scheduling
17.	SG	Shortest Gap
18.	PMW	Periodic Min-Max Algorithm
19.	ACO	Ant Colony Optimization
20.	PSO	Particle Swarm Optimization
21.	DoI	Degree-of-Imbalance
22.	FCFS	First Come First Serve
23.	GSA	Gravitational Search Algorithm
24.	CSP	Cloud Service Provider
25.	CIS	Cloud Information Service
26.	ALS	Adaptive Local Search
27.	EPL	Experience-based Perturbed Learning
28.	DoF	Degree-of-Freedom
29.	ML	Machine Learning
30.	NP	Non-Polynomial

- Transforms the continuous solution using a binary version of the IARO-PS method into a discrete solution,
- Analyzes the statistical significance of the algorithms employing the Friedman test, and
- Evaluates the proposed IARO-PS algorithm's efficacy against several QoS metrics for both homogeneous and heterogeneous resources using a real-world workload from NASA in CloudSim. It also makes comparisons with various load balancing and task scheduling algorithms, such as GA, BSO, QMPSO, and MGWO.

The sections that follow are organized methodically. The literature on various scheduling and load-balancing strategies is reviewed in *Section II*. The Cloud scheduler model and the related problem formulations are shown in *Section III*. The suggested IARO-PS load balancing technique is explained in *Section IV* by emphasizing the flowcharts. The experimental setup and the simulation findings are highlighted in *Section V*. An account of the results is given in *Section VI*. Finally, *Section VII* provides a summary of the findings and some implications for the future.

II. REVIEW WORKS

Every static as well as dynamic load balancing method has certain drawbacks. Combining these with other metaheuristic strategies is the most effective means to overcome these restrictions because it allows us to combine the benefits of different algorithms to overcome their shortcomings. A hybrid algorithm combines two algorithms into one. There are hybrid strategies in the literature that combine heuristics and metaheuristic algorithms with other strategies to overcome their shortcomings. These hybrid approaches eliminate those inherent shortcomings while demonstrating a performance and efficiency that is noticeably better. The researchers' related works on cloud-based system hybridization approaches have been presented. Table 2 presents the related literature works highlighting the contributions and limitations of the existing approaches.

A new Cloud job scheduling technique called SG-PBFS has been proposed by Murad et. al. [12] to tackle the challenges of dynamic Cloud environments. SG-PBFS combines prioritizing jobs by using PBFSwhile fitting them into the smallest available gaps whereShortest Gap strategy is utilized. While simulations show SG-PBFS outperforms existing algorithms in terms of completion time and tardiness, the research doesn't address how it handles interrupting jobs for higher priorities (*i.e.*, pre-emption). Additionally, real-world testing would strengthen the understanding of SG-PBFS effectiveness in Cloud environments.

To overcome the load balancing problem, Jena et. al. [23] finally created a hybrid mechanism in conjunction with the modified PSO and Q-learning technique. To gauge the fitness of the particles, a fitness function is used, taking load-balancing and power consumption into consideration. The resources used in simulations are uniform. To verify the algorithm's efficacy, however, diverse resources and real-world datasets ought to have been used. Given their importance in load balancing, the makespan and VM utilization may have been included in the fitness function to accomplish the load balancing goal. This technique is being contrasted with the present one.

The present approach focuses on combining PS with a hybrid task scheduling technique called IARO. The low computational complexity of the IARO algorithm overcomes the constraint of PS. PS is used to generate the first solution. Nevertheless, those tasks are mapped onto the appropriate VMs via IARO.

III. PROPOSED CLOUD FRAMEWORK AND PROBLEM DEFINITIONS

Table 3 denotes all the symbols used in this research article along with their respective descriptions.

A. CLOUD SCHEDULING FRAMEWORK

As shown in Figure 1, let us consider a cloud scheduling paradigm that consists of numerous VMs running on multiple hosts. The entities that make up this architecture are as follows:

- **Cloud user:** This is the customer requesting cloud facilities to request that a broker or supplier of cloud services assist them in carrying out specific tasks.
- **Cloud broker:** It is also known as CSP. The CSP uses a pay-as-you-go pricing model to provide its

TABLE 2. Literature review.

Sl. No.	References	Contributions	Limitations
1.	Murad*et. al.*(2024) [12]	• The paper proposes a new scheduling technique called SG-PBFS for job scheduling in Cloud environments. • SG-PBFS combines two techniques: PBFS and a backfilling strategy called SG. • PBFS assigns priorities to tasks, while the SG strategy tries to fit tasks into the smallest available gaps in the schedule. • The authors claim that SG-PBFS outperforms other scheduling algorithms in terms of flow time, makespan time, and total tardiness.	• The paper does not mention how SG-PBFS handles pre-emption (*i.e.*, interrupting a running task to allow a higher-priority task to run).
2.	Banerjee*et. al.*(2023) [13]	• Introduces MTD-DHJS, a makespan-optimized task scheduling algorithm designed specifically for Cloud computing environments. • Incorporates dynamic computational time prediction, allowing for more accurate scheduling decisions in dynamic Cloud environments where task execution times may vary. • Aims to minimize makespan, thereby improving overall system efficiency and resource utilization. • Utilizes an optimization approach to efficiently allocate tasks to available resources in the Cloud, considering variations in computational time. • Provides potential improvements in Cloud computing performance, particularly in terms of task scheduling efficiency and overall system throughput.	• The effectiveness of the proposed algorithm heavily depends on the accuracy of the dynamic computational time prediction mechanism, which could introduce uncertainties in real-world scenarios. • There might be challenges in extending the algorithm's applicability to large-scale Cloud environments with diverse workload patterns and resource characteristics. • The scalability and robustness of the algorithm in handling various workload scenarios and system dynamics remain to be thoroughly evaluated.
3.	Murad*et. al.*(2024) [14]	• The authors proposed a new job scheduling technique called PBFS specifically designed for Cloud computing environments. • PBFS incorporates task priorities into the scheduling process, potentially ensuring important tasks are completed first. • The technique leverages gaps in the resource schedule to fit in smaller tasks, potentially improving resource utilization. • The authors claim PBFS achieves better results in terms of overall delay, makespan, and flow time compared to other existing scheduling algorithms.	• The effectiveness of the proposed technique may depend on the accuracy of task prioritization criteria and the ability to identify and exploit gaps in task scheduling effectively. • The scalability and robustness of the technique in handling large-scale task scheduling scenarios and dynamic workload fluctuations remain to be thoroughly evaluated.
4.	Sharma*et. al.*(2023) [15]	• The research proposes using a cloud-based platform for multi-robot task scheduling. This could offer benefits like centralized control, scalability, and access to powerful computing resources for complex scheduling tasks. • The paper introduces the PMW specifically for multi-robot task scheduling which considers factors like task completion time and robot load balancing to create an efficient schedule. • The authors claim that using PMW in a Cloud-based system leads to improvements in task completion time and load balancing compared to traditional scheduling approaches.	• The scalability and robustness of the approach in handling large-scale multi-robot task scheduling scenarios and dynamic workload fluctuations remain to be thoroughly evaluated. • The effectiveness of the proposed approach may depend on the accuracy of task prioritization criteria and the ability to efficiently manage robot mobility and task waiting times.
5.	Cho *et. al.*(2015) [16]	• Presenting a hybridized metaheuristic approach by combining PSO with ACO. • Utilizing PSO with ACO to overcome the disadvantage of ACO becoming trapped in local optima. • Better results in processing time, DoI, highest completion time, and response time.	• Drawbacks in fault tolerance, energy consumption, and dependability.
6.	Shojafar*et. al.*(2015) [17]	• Combining fuzzy theory with GA for load dispersion. • Considering VM's memory, processing speed, bandwidth, and task duration. • Increase in DoI, cost, processing time, and makespan.	• Inadequate addressing of power consumption and migration time.
7.	Dam *et. al.*(2015) [18]	• Developed a hybrid method for load balancing by combining GA with gravitational local search. • Maximizing VM utilization by considering VM's idle time. • Outperforming ACO and FCFS in terms of makespan, reaction time, and QoS.	• Drawbacks in energy usage, fault tolerance, and dependent job consideration.
8.	Jeyakrishnan*et. al.*(2017) [19]	• Presented an approach modelled after PSO and Bacterial Swarm Optimization for task scheduling.Strengthening each other's shortcomings. • Performed better in makespan, operational costs, reaction times, and VM usage.	• Did not address scalability and low throughput adequately.
9.	Wei *et. al.*(2017) [20]	• Proposed a method combining multi-population GA with SA for task scheduling. • UtilizedMax-Min algorithm for initial population generation. • Demonstrated improved outcomes in makespan, cost, and completion time.	• Significant drawbacks in VM usage and power usage.

TABLE 2. *(Continued.)* Literature review.

10.	Alla*et. al.*(2017) [21]	• Created a unique variable priority queue called ELECTRE III employing SA and PSO for VM assignment. • Reducedmakespan and maximizing VM utilization by prioritizing tasks based on length, deadline, completion time, and waiting time.	• Not adequately testing with diverse resources and real-world datasets.
11.	Rani *et. al.*(2020) [22]	• Presented a method combining GSA with ACO for load balancing. • Eliminated ACO's vulnerability to local optima by using GSA. • Significantly reducedmakespanand increased VM utilization while demonstrating efficacy in throughput, scalability, and load balancing.	• Overhead costs involved.
12.	Jena *et. al.*(2022) [23]	• Created a hybrid mechanism with modified PSO and Q-learning technique for load balancing. • Utilized fitness function considering load-balancing and power consumption.	• Not adequately testing with diverse resources and real-world datasets. • Importance of makespan and VM utilization not fully incorporated in fitness function.

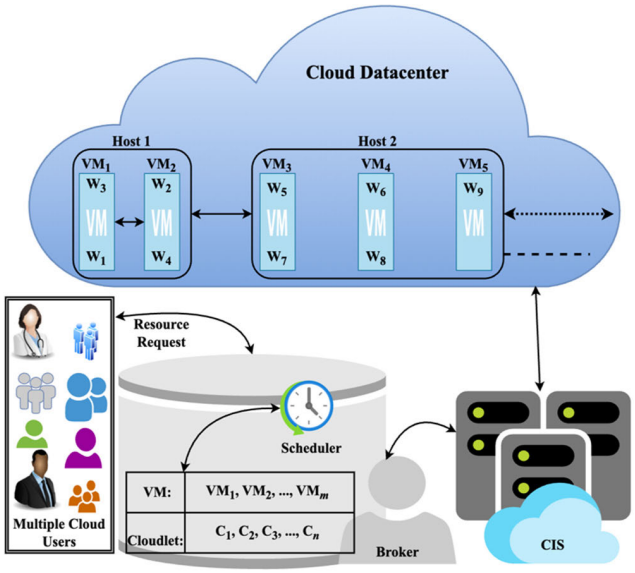

FIGURE 1. Scheduling model in cloud.

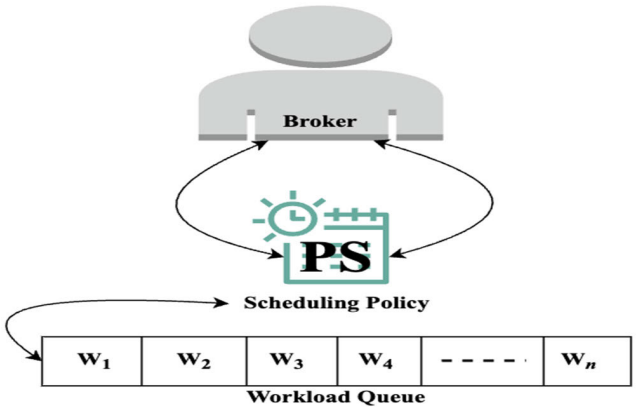

FIGURE 2. IARO-PS-based scheduling framework.

customers with on-demand services. They use several VMs deployed on fewer hosts to provide services. In this case, one cloud/datacenter is assumed. In consultation with a CIS, the broker forwards requests during peak demand to any additional cloud or any other underutilized hosts located within the same cloud/datacenter. It is also in charge of assigning user tasks to VMs so they can be executed under a schedule.

- **Cloud Information Service (CIS):** It is a storehouse which includes every crucial feature of the component, including the host, scheduling policy, VM, cloudlet (task), and so forth. Before allocating any work to an underlying VM, the broker confers with the CIS after it has arrived. The CIS stores the attributes of each task.

These elements oversee handling task assignments which are described as follows. The broker verifies the availability of VMs and maintains the workloads in a waiting queue when a user requests that they be executed. The broker assigns jobs to VMs that are compatible and have a minimal completion time requirement. The workload moves on to the host's ready queue, where VMs are housed after a suitable VM has been found. According to how each VM is used, the broker assigns all requests that arrive to the appropriate VMs in this manner. The broker attempts to divide up all incoming loads across the VMs so that no VM is left idle or overloaded. According to a scheduling strategy that must be created to minimize the makespan by efficiently using the VMs, the broker allocates the work to the appropriate VMs. As seen in Figure 2, we suggest an IARO-based scheduling system. The IARO approach will be used by the broker to schedule tasks. The status of load imbalance would be resolved by assigning jobs to VMs.

B. TASK-RESOURCE PROBLEM FORMULATIONS

Definition 1 (Datacentre): It is the cloud architecture made up of virtual machines running on physical hosts. For the simulation work, we take into consideration one datacenter in this research. Another name for it is cloud.

Definition 2 (Host Lists): These are numerous servers kept in a cloud to plan out user requests for services.It is identified as $HO = \{HO_1, HO_2, HO_3, \ldots, HO_h\}$.

Definition 3 (VM Set): Factor in a set of $R = \{R_1, R_2, R_3, \ldots, R_m\}$ where R_j, $1 \leq j \leq m$ is a set of VMs hosted under numerous servers. A $R_j \in R$, $1 \leq j \leq m$ has the processing speed P_{jr} in MIPS.

Definition 4 (Task Set): Take into account a task set $W = \{W_1, W_2, W_3, \ldots, W_n\}$ of dynamic and non-pre-emptive workloads, where W_i, $1 \leq i \leq n$ is the i^{th} task with the

TABLE 3. List of notations used.

Sl. No.	Symbols/Notations	Descriptions
1.	$HO = \{HO_1, HO_2, HO_3, \ldots, HO_h\}$	Host list
2.	$R = \{R_1, R_2, R_3, \ldots, R_m\}$	Set of VMs
3.	$W = \{W_1, W_2, W_3, \ldots, W_n\}$	Set of tasks
4.	$ST(W_j)$	Task's start time
5.	$FT(W_j)$	Task's finish time
6.	$T_{CT_{ij}}$	Completion time of a task W_i on a VM R_j
7.	$max\left(T_{CT_{ij}}\right)$	Total time for execution of task W_i
8.	MS	Makespan
9.	$RU_{R_{ij}}$	Resource utilization of a VM R_j by a workload W_i
10.	$AvgRU_{R_{ij}}$	Average resource utilization
11.	fn_{val}	Fitness function in terms of $AvgRU_{R_{ij}}$and MS
12.	T_{max}	Maximum processing time
13.	T_{min}	Minimum processing time
14.	T_{mean}	Mean completion time
15.	L	Length of total instruction
16.	$num(PNs)_{R_j}$	Set of processing elements in the j^{th} VM
17.	$MIPS(PNs)_{R_j}$	Million instructions per second of the j^{th} VM
18.	DoI	Degree-of-Imbalance
19.	P_j	Potential of a VM
20.	$B(R_j)$	Bandwidth of a VM
21.	P	Overall capacity of a VM
22.	$Wl(R_j)$	Load on a VM
23.	Wl	Mean system load on a server and the workload of all VMs
24.	$AvgLoad$	Average workload of the system
25.	NV	Normalized VM
26.	OV	Overloaded VM
27.	UV	Underloaded VM
28.	$T_{R_{usd}}(UV_j)$	Total resources used by tasks of an underloaded VM
29.	$T_{R_{avl}}(UV_j)$	Availability of underloaded resources
30.	T_R	Total resources
31.	T_K	Overloaded VMs' workloads
32.	θ	Compatibility
33.	η	Cosine similarity
34.	μ	constant
35.	T_{sh}	Threshold value
36.	σ	Standard deviation
37.	PT_{ij}	Execution time of a task i on j^{th} VM
38.	PT	Total VM execution time
39.	$\overline{z_i}(t)$	Candidate's recent position of the i^{th} candidate at time t

TABLE 3. *(Continued.)* List of notations used.

40.	$\overrightarrow{\Delta_i}(t+1)$	i^{th} updated candidate at time $t+1$
41.	M	Total number of iterations
42.	T	Size of the rabbit population
43.	d	The problem dimension or creating burrows inside the rabbit's territory
44.	$randperm$	The random permutation function that spanned from 1 to the problem dimension
45.	E	The running length during foraging
46.	n_1	Function of the normal distribution
47.	g_1, g_2, g_3, g_4, g_5	Consistent random values between [0, 1]
48.	c	A vector
49.	ρ	A mathematical function that mimics rabbit movements
50.	H	Hiding function
51.	$BU_{ir}(t)$	Burrow that the rabbit has selected using the hiding mode
52.	$EA(t)$	Energy of the rabbit
53.	α	Random integer
54.	Δ_{mean}^{it}	Mean of a randomly selected solution
55.	Δ_{dev}^{it}	Deviation of a randomly selected solution
56.	Δ_{best}	Best solution identified thus far in the EPL process
57.	$rand_1, rand_2$,and$rand_3$	Three random integers elicited accordance to the uniform distribution within the interval [0,1]
58.	Δ_I^{it}	Any arbitrary solution picked at random
59.	Δ_{max}andΔ_{min}	Dynamical bounders
60.	z_{worst}	Worst individual
61.	z_{best}	Best individual
62.	z_I^{it}	Present solution of the i^{th} solution inside the elite class
63.	z_I^{it+1}	Updated solution of the i^{th} solution inside the elite class
64.	$I0$	Initial point of PS
65.	NH_0	Null hypothesis
66.	R_i	Friedman average rank
67.	k	Collection of methodologies
68.	P	Collection of performance parameters
69.	F_F	Friedman statistics

million sets of instruction (MI) as I_i.In this case, the set of real-world workloads is handled like a collection of tasks to be performed on several VMs.

Definition 5 (QoS): It serves as a gauge to determine how successful any algorithm is. The services are provided based on these metrics. Therefore, before using the services, clients must enter into an SLA-based agreement with the CSP regarding this QoS. To fulfil the requirements of the customers promptly and without delay, it deals with

the throughput, response time, VM utilization, processing time, power usage, scalability, and availability. The QoS metrics taken into consideration in this work include VM utilization, response time, DoI, and makespan.

Definition 6 (Completion Time): It is the amount of time needed by a R_j to complete carrying out a task W_i. It is stated as the difference between a workload's start time $\left(ST(W_j)\right)$ and finish time $\left(FT(W_j)\right)$ by a R_j indicated as T_j. The workload completion time ($T_{CT_{ij}}$) is defined in Eq.(1).

$$T_{CT_{ij}} = FT\left(W_j\right) - ST(W_j) \tag{1}$$

Definition 7 (Makespan): It specifies the total amount of time spent on execution $\left(max\left(T_{CT_{ij}}\right)\right)$ by a task W_i among the VMs. For scheduling to be effective, this time must be shortened. It is defined using Eq.(2) [24], [25].

$$MS = max\{T_{CT_{ij}} | i = 1, 2, \ldots, n; j = 1, 2, \ldots, m\} \tag{2}$$

Definition 8 (Utilization): It speaks of the extent to which VMs are used [26], [27]. It is a major contributing factor in active load balancingstrategy. The relationship between makespan and VM utilization is inversely proportional to each other and is as follows:

$$RU_{R_{ij}} = \frac{T_{CT_{ij}}}{MS} \tag{3}$$

Average resource utilization is calculated using Eq. (4).

$$AvgRU_{R_{ij}} = \frac{\sum_{j=1}^{m} U_{R_{ij}}}{m} \tag{4}$$

where, m is the total set of VMs.

Definition 9 (Fitness Function): It is the procedure for assigning and planning tasks to the VMs in a cloud setting. This work employs a fitness function that maximizes resource utilization while being subject to a loss in makespan. Eq. (5) defines the fitness function (fn_{val}) in terms of the average resource utilization and makespan. A lower fitness value, according to Eq. (5), suggests that a task is in the optimal location. Thus, it is shown as:

$$fn_{val} = \frac{1}{MS} \times AvgRU_{R_{ij}} \tag{5}$$

The challenge of task scheduling is expressed as follows:consider a number of tasks W and a number of VMs R. The issue is how to assign W to the underlying VMs ($fn : W \rightarrow R$) generated over several hosts with various technical setups via the cloud service broker to reduce the overall completion time (*i.e.*, makespan) and increase VM utilization.

1) DEGREE OF IMBALANCE (DOI)

It operates on the following Eq. (6) and Eq. (7) [29] and determines the workload imbalances across VMs [28]. Where, T_{max} means maximum processing time, and T_{min} meansa minimum processing time of task T_i amidst VMs. And T_{mean} is the meancompletion time of tasks. L indicatesthe length of total instruction, $num\ (PNs)_{R_j}$ indicates the set of processing elements in the j^{th} VM, and $MIPS\ (PNs)_{R_j}$ is the million instructions per second of the j^{th} VM.

$$DoI = \frac{T_{max} - T_{min}}{T_{mean}} \tag{6}$$

$$T_i = \frac{L}{num\ (PNs)_{R_j} \times MIPS\ (PNs)_{R_j}} \tag{7}$$

2) POTENTIAL OF A VM

The VM's potential $\left(P_j\right)$ is defined using the following equation [47].

$$P_j = num\ (PNs)_{R_j} \times MIPS\ (PNs)_{R_j} \times B\left(R_j\right) \tag{8}$$

where $B\left(R_j\right)$ is the bandwidth of a VM.

The overall VMs' capacity (P) is expressed as follows.

$$P = \sum_{j=1}^{m} P_j \tag{9}$$

3) WORKLOAD OF A VM

We must gauge the loads on each VM and arrange them based on the loads to trade off across VMs. It can be defined as the ratio of a VM's processing time to the total number of tasks allocated to it at a given time t [24], [30]. Thus, Eq. (10) and Eq. (11) can be used to determine both the burden on a VM and the overall workload.

$$Wl\left(R_j\right) = \frac{NT(W, t)}{PT(R_{ij}, t)} \tag{10}$$

The mean system load on a server and the workload of all VMs are computed as:

$$Wl = \sum_{j=1}^{m} L(VM_j) \tag{11}$$

$$AvgLoad = \frac{1}{m} \sum_{j=1}^{m} Wl(R_j) \tag{12}$$

4) STATE OF THE VM GROUP

To determine the status of every VM, the workload of each VM $\left(R_j\right)$ is compared to the average workload of the system ($AvgLoad$). This will determine if the VM's state is normalized (NV), overloaded (OV), or underloaded (UV).

$$if \begin{cases} Wl\left(R_j\right) < AvgLoad \rightarrow UV \\ Wl\left(R_j\right) > AvgLoad \rightarrow OV \\ Wl\left(R_j\right) = AvgLoad \rightarrow NV \end{cases} \tag{13}$$

5) TASK MIGRATION PROCEDURE

The broker locates the VM group and then starts the process of moving the workload from the OV to the UV. To map the jobs onto UV, it would be beneficial to verify the complete available resources at the UV before migration. Eq. (14) and Eq. (15) are used to estimate it [31].

$$T_{R_{usd}}\left(UV_j\right) = \sum_{i=1}^{n} T_i \tag{14}$$

$$T_{R_{avl}}\left(UV_j\right) = T_R - T_{R_{usd}} \tag{15}$$

where, $T_{R_{usd}}\left(UV_j\right)$ is the total resources usedby tasks of an underloaded VM, $T_{R_{avl}}\left(UV_j\right)$ is the availability of underloaded resources and T_R is the total resources. Assuming 100% of the available resources are being utilized.

Underloaded VMs' resources that are available are shown by $T_{R_{avl}}\left(UV_j\right)$, and overloaded VMs' workloads are represented by T_K. Prior to migration, compatibility (θ) is to be calculated between T_K and $T_{R_{avl}}\left(UV_j\right)$ by a process known as cosine similarity (η). Makespan, the utilization of resources, and the value of η are used to convey it [32]. A larger degree of commonality among workloads T_K and $T_{R_{avl}}\left(UV_j\right)$ is present when the value of η is less. Therefore, in this article, the value of μ is fixed to 0.5 and the job having the highest degree of compatibility for a UV is chosen. η and θ are evaluatedby Eq.(16) and Eq.(17):

$$\eta = cos^{-1}\left(\frac{T_K \times T_{R_{avl}}\left(UV_j\right)}{|T_K|\left|T_{R_{avl}}\left(UV_j\right)\right|}\right) \tag{16}$$

$$\theta = \mu \times \eta + (1-\mu) \times RU_{VM_{ij}} \tag{17}$$

6) IDENTIFYING THE LACK OF LOAD DEVIATION

It is necessary to compute the load's standard deviation to determine the overall system's degree of balance. If the standard deviation's value is within the acceptable range of threshold value $(T_{sh}[0-1])$, the system is called balanced. Because a system's average system load shouldn't be greater than its maximum capacity, the threshold value depends on the system's maximum capacity. Without a doubt, the criterion T_{sh} for a fully utilized system is 80% or 90% [33]. TS_{sh} is therefore set to the value 0.9.Eq. (18) is used to estimate it where the execution time of a task i on j^{th} VM is PT_{ij} andthe total VM execution time is denoted by PT.

$$\sigma = \sqrt{\frac{1}{m}\sum_{j=1}^{m}(PT_{ij} - PT)^2} \tag{18}$$

$$\text{where, } PT_{ij} = \frac{Wl(R_j)}{AvgLoad} \tag{19}$$

$$PT = \sum_{j=1}^{m} PT_{ij}$$

$$if \begin{cases} \sigma \leq T_{sh}\,(0-1) \rightarrow Balanced \\ otherwise \rightarrow Imbalanced \end{cases} \tag{20}$$

IV. THE HYBRID IARO-PS-BASED TASK SCHEDULING

The regular ARO method, IARO, and a binary version of the IARO-PS approach are all briefly outlined in this section. Furthermore, it provides a comprehensive understanding of the suggested IARO-PS data scheduling methodology by emphasizing the logical process.

A. ARTIFICIAL RABBIT OPTIMIZATION (ARO)

The ARO algorithm is a novel and potent metaheuristic method that draws inspiration from rabbit survival strategies [49] such as randomized sheltering and detour foraging. The algorithm models the foraging mode, where rabbits attempt to eat plants near other burrows to deceive predators and protect their burrows. Instead of consuming nearby food, rabbits search for it in distant locations. The swarm population size in ARO is determined by the number of rabbits, and every rabbit has a grassy and plant-filled feeding area, as well as several burrows. During foraging, rabbits randomly visit other rabbits' burrows for food, updating their locations based on the chosen individual and introducing disruptions. This foraging activity can be mathematically described as follows:

$$\vec{\Delta}_i\,(t+1) = \vec{z}_j\,(t) + \rho.\left(\vec{z}_i\,(t) - \vec{z}_j\,(t)\right) + round\,(0.5.\,(0.05{+}g_1))\,.n_1, i, \quad j = 1, 2\ldots\ldots, Mandi \neq j \tag{21}$$

$$\rho = E.c \tag{22}$$

$$E = \left(e - e^{\left(\frac{1-t}{T}\right)^2}\right).\sin(2\pi g_2) \tag{23}$$

$$c\,(k) = \begin{cases} 1 & if\ k == h\,(u) \\ 0 & else \end{cases}, k = 1, \ldots, d\,\&\, u = 1, 2, \ldots, \lceil g_3.d \rceil \tag{24}$$

$$h = randperm(d) \tag{25}$$

$$n_1 \sim N(0, 1) \tag{26}$$

where $\vec{z}_i\,(t)$, $\vec{\Delta}_i\,(t+1)$, M, T, d, $randperm$, and E describes the candidate's recent position of i^{th} candidate at time t, i^{th} updated candidate at time $t+1$, the total number of iterations, the size of the rabbit population, the problem dimension, the random permutation function that spanned from 1 to the problem dimension, and the running length during foraging, in that order. Here n_1 signifies the function of the normal distribution, and g_1, g_2, g_3 specify consistent random values between [0, 1]. The variation in Eq. (26) helps ARO perform a thorough search and avoid local maxima/minima. To carry out the exploration and exploitation modes, respectively, the running distance (E) aids in navigating longer distances in the initial iterations and less distances in the subsequent iterations.Here, c is a vector that is utilized in the process of searching to choose many individuals, and ρ is a mathematical function that mimics rabbit movement. Consequently, the ARO algorithm's exploration and global identification ability are enhanced during this foraging stage.

To avoid predators, rabbits use a haphazard concealment technique during the exploitation phase. They dig a few holes close to where they started. A rabbit constantly produces b burrows in every dimension throughout an ARO iteration. Then, it chooses one of these holes at random to conceal itself from predators.The i^{th} rabbit with the j^{th} burrow can be expressed mathematically in the following way:

$$\vec{BU}_{ij}\,(t) = \vec{z}_i\,(t) + H.h.\vec{z}_i\,(t)\,, i = 1, 2\ldots\ldots, \quad Mandj = 1, 2\ldots\ldots, d \tag{27}$$

$$H = \frac{1 - t + T}{T} g_4 \tag{28}$$

$$n_2 \sim N(0, 1) \tag{29}$$

$$h(k) = \begin{cases} 1 & if\ k == j \\ 0 & else \end{cases}, k = 1, \ldots, d \quad (30)$$

where, H stands for the hiding function and d stands for creating burrows inside the rabbit's territory. A rabbit's large region is mostly where holes are made. As the number of iterations increases, the neighborhood's size decreases. The following is the available random concealing mode:

$$\vec{\Delta}_i(t+1) = \vec{z}_i(t) + \rho.\left(g_4.BU_{ir}(t) - \vec{z}_i(t)\right), \quad i = 1, 2, \ldots\ldots M \quad (31)$$

$$h_r(k) = \begin{cases} 1 & if\ k == [g_5.d] \\ 0 & if\ k == [g_5.d] \end{cases}, k = 1, \ldots, d \quad (32)$$

$$BU_{ir}(t) = \vec{z}_i(t) + H.h_r.\vec{z}_i(t) \quad (33)$$

where, $BU_{ir}(t)$ indicates the burrow that the rabbit has selected using the hiding mode; g_4 and g_5 specify random integers that fall between 0 and 1. Following either a random concealment procedure or a detour foraging mode, the position of the i^{th} rabbit is updated as follows:

$$\vec{z}_s(t+1) = \begin{cases} \vec{z}_s(t) & f(\vec{z}_s(t)) \leq f(\vec{\Delta}_s(t+1)) \\ \vec{\Delta}_s(t+1) & f(\vec{z}_s(t)) > f(\vec{\Delta}_s(t+1)) \end{cases} \quad (34)$$

The rabbit leaves its present spot and remains in the candidate location indicated by either Eq. (21) or Eq. (31) if the s^{th} rabbit's candidate fitness is greater than the position's current fitness. As repetition progresses, rabbit energy decreases, aiding in the shift from exploratory to exploitative mode, which is expressed as follows:

$$EA(t) = 4(1 - \frac{t}{T})\ln(\frac{1}{\alpha}) \quad (35)$$

where a random integer is defined by α. The method looks locally for the solution (exploitation) when $EA(t) \leq 1$, and globally for the solution (exploration) when $EA(t) > 1$.

1) IMPROVED VERSION OF ARTIFICIAL RABBITS OPTIMIZATION (IARO)

This work discusses an enhanced version of the ARO method, called IARO. ARO has shortcomings when it comes to tackling complex issues, but it has benefits in simplicity and quick convergence. ALS and EPL techniques are used by IARO to overcome these problems. While EPL strikes a balance between exploration and exploitation, ALS assists in concentrating on prospective locations while avoiding local optima. IARO exhibits improved efficacy overall for optimization tasks.

Rabbits follow one another in the population as part of a novel exploration technique introduced by the IARO algorithm. This revised plan, therefore, may result in invasive diversification patterns. To improve exploration and find more attractive regions inside the feasible search space, the algorithm integrates the EPL technique. First, the mean (Δ^{it}_{mean}) and deviation (Δ^{it}_{dev}) of a randomly selected solution are computed about the best solution(Δ_{best}) identified thus far in the EPL process. After that, a perturbed term and these data are used to update the new solution. The following is an expression for the updated solution:

$$\Delta^{it}_{mean} = (z_{best} + z^{it}_I)/2 \quad (36)$$

$$\Delta^{it}_{dev} = \text{abs}(z_{best} - z^{it}_I) \quad (37)$$

$$\Delta^{it}_C = \Delta^{it}_{mean} + rand_1.\Delta^{it}_{dev} \quad (38)$$

$$z^{it}_{new} = \Delta^{it}_C + rand_2.(z_{best} - \Delta^{it}_C) + 0.95^{it} .(rand_3 - 0.5).\text{abs}(z_{max,j} - z_{min,j}), z_{max,j} = max_j\left\{z^{it}_i\right\}, z_{min,j} = min_j\left\{z^{it}_i\right\} \forall \quad (39)$$

where $rand_1$, $rand_2$, and $rand_3$ defines three random integers elicited accordance to the uniform distribution within the interval [0,1], and Δ^{it}_I stands for any arbitrary solution picked at random. In this case, perturbed solutions inside the dynamical bounders ($\Delta_{max} and \Delta_{min}$) are performed using the third element of Eq. (39). By incorporating Exploration and Exploitation into the IARO algorithm, EPL improves search performance and accelerates converging to optimal solutions.

We suggest an ALS technique to increase accuracy through the process of iteration and encourage exploitation within the promising space. This guiding approach makes use of shared data between the top rabbit group and their worst individual (z_{worst}) and best individual (z_{best}). First, using the fitness function, we identify the elite group and, within it, the worst and best individuals (z^B and z^W). The updating phase then uses three different sorts of movements: (1) pushing z^W in the direction of z^B, (2) pushing z^W in the direction of z_{best}, and (3) pushing z^W in the way of the average of z^B and z_{best}. These exercises are done one after the other until one of them achieves a higher level of fitness. This strategy's update phase can be stated as follows:

$$z^{it+1}_I = \begin{cases} z^{ALS}_1 = 2 \times r_1 \times \left(z^B - z^W\right) + z^W \\ \quad iff\ (z^{ALS}_1) \leq f(z^{it}_I) \\ z^{ALS}_2 = 2 \times r_2 \times \left(z_{best} - z^W\right) + z^W \\ \quad elseiff\ (z^{ALS}_2) \leq f(z^{it}_I) \\ z^{ALS}_3 = 2 \times r_3 \times \left(\left(z^B + z_{best}\right) \Big/ 2 - z^W\right) + z^W \\ \quad otherwise \end{cases} \quad (40)$$

where z^{it}_I and z^{it+1}_I stand for the present and updated solutions of the i^{th} solution inside the elite class.

2) THE PATTERN SEARCH METHOD

It is a derivative-free evolutionary optimization approach that is preferred for resolving a variety of optimization issues that fall outside the purview of conventional optimization methods. Due to some of its advantages over other evolutionary techniques, such as conceptual simplicity, ease of implementation, and effective computational efficiency, this algorithm is well-liked among optimization experts. It's an appropriate optimization technique with a flexible and well-balanced

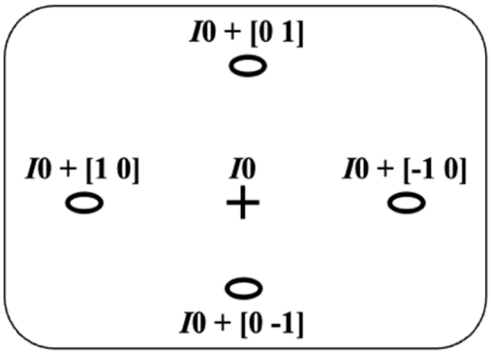

FIGURE 3. Pattern Search mesh points with patterns.

operator to improve and adjust global search and fine-tune local search [50]. The PS method typically computes a series of points that might or might not result in the optimal point. The basic first step begins with a mesh of points surrounding the initial points. Such starting points, or current points, are provided by the IARO approach. The current point is then added to a scalar multiple of a pattern, which is a collection of vectors, to create the mesh. The mesh point with the modified objective function value is chosen as the current point for the subsequent iteration.

The PS begins at the initial point ($I0$), which is the result of the IARO algorithm. In the first iteration, the direction vectors or pattern vectors are created as [01], [10], [−10] and [0 − 1] for the mesh size with scalar = 1. Next, as shown in Figure3, the summation of the starting point $I0$ and pattern vectors for calculating the mesh points as $I0+[01]$, $I0+[10]$, $I0+[-10]$ and $I0 + [0 - 1]$. The approach computes the goal function in the same order at the mesh locations. The method polls the mesh points by computing their cost function values until it finds one value that is smaller than the cost function value of $I0$.

If the objective function's value decreases at a few mesh points and the algorithm designates this point as $I1$, the poll will be considered successful in this case. The algorithm will advance to the next iteration and multiply the current mesh size by two, known as the expansion factor, after each successful result. The second iteration repeats the process until the halting criteria are satisfied, with the mesh points being $I1+2*[01]$, $I1+2*[10]$, $I1+2*[-10]$ and $I1+2*[0-1]$. The current point is passed to a subsequent iteration for better usage if the poll is deemed unsuccessful at any iteration due to a smaller objective function value. In this iteration, the algorithm multiplies a contraction factor of 0.5 to the current mesh size, which is reduced until the stopping condition is met. Figure 4 shows the suggested IARO-PS method flow chart.

B. THE BINARY IARO-PS ALGORITHM

The IARO-PS is not appropriate to describe the task-resource allocations in cloud computing since it is meant to handle the restricted and unconstrained optimization issue, which yields continuous solutions. These have since been extended to deal with binary value-only optimization difficulties. It must be expressed in the form of 0s and 1s, just as the scheduling issue is in our method. Like how scheduling problems are assigned in cloud computing, in binary IARO-PS the particles are moving in a limited direction that is limited to either 0 or 1. The resulting continuous solutions should be transformed into discrete solutions since the representation of the task-resource assignment depends on binary values. Therefore, the tangent hyperbolic logistic transfer function [40], [41] is used in this work to translate generated continuous values into discrete values that are represented by Eq. (41) and Eq. (42).

$$tanh\left(\left|X_i^{k+1}\right|\right) = \frac{e^{\left(\left|2X_i^{k+1}\right|\right)} - 1}{e^{\left(\left|2X_i^{k+1}\right|\right)} + 1} \tag{41}$$

Using Eq. (42), the binary solution for the revised value of X_i^{k+1} is produced.

$$X_i^{k+1} = \begin{cases} 1, & if\ rand() < \tanh\left(\left|X_i^{k+1}\right|\right) \\ 0, & otherwise \end{cases} \tag{42}$$

V. EXPERIMENTAL SETUP AND SIMULATION RESULTS

This work's experimental configuration can be divided into two test scenarios. In the first example, MATLAB is used to assess the suggested IARO-PS algorithm's effectiveness against a benchmark function. The outcome is then displayed. In the second trial scenario, CloudSim [34], [35] is used to evaluate the effectiveness of the suggested IARO-PS for a variety of QoS scheduling metrics on a baseline dataset that includes both homogeneous and diverse VMs.

A. TRIAL CASE 1: EFFECTIVENESS OF HYBRID IARO-PS TECHNIQUE IN MATLAB

Using many standard benchmark functions (CEC2017), the effectiveness of the developed IARO-PS is assessed, and the fitness values' worst, best, standard deviation, and mean are determined. The simulation runs in MATLAB *R2023a* with an Intel i7 processor running Windows 11 and 8 GB of RAM. We ran 100 runs in all, taking into account 50 dimensions for every function. The region of the search space that is being evaluated is [−100, 100]. We used the Friedman rank test to create ranks and comparisons. The assessment results of the QMPSO [23], GA, and MGWO [42] algorithms are contrasted with the IARO-PS algorithm. Tables 4 and 5 list the parameters for the algorithms for IARO-PS and the computational index for BSO, QMPSO, GA, and MGWO. The simulation's results are listed in Table 6 (*refer Appendix*), and Table 7 (*refer Appendix*) displays the average rank of the suggested hybrid technique and different methodologies discussed according to the CEC2017 baseline function employing Friedman's rank test.

B. TRIAL CASE 2: EFFECTIVENESS OF HYBRID IARO-PS FOR SCHEDULING IN CLOUDSIM

To model our simulation, we have extended the classes in CloudSim. The *datacenterBroker* class's *bindCloudletToVM*

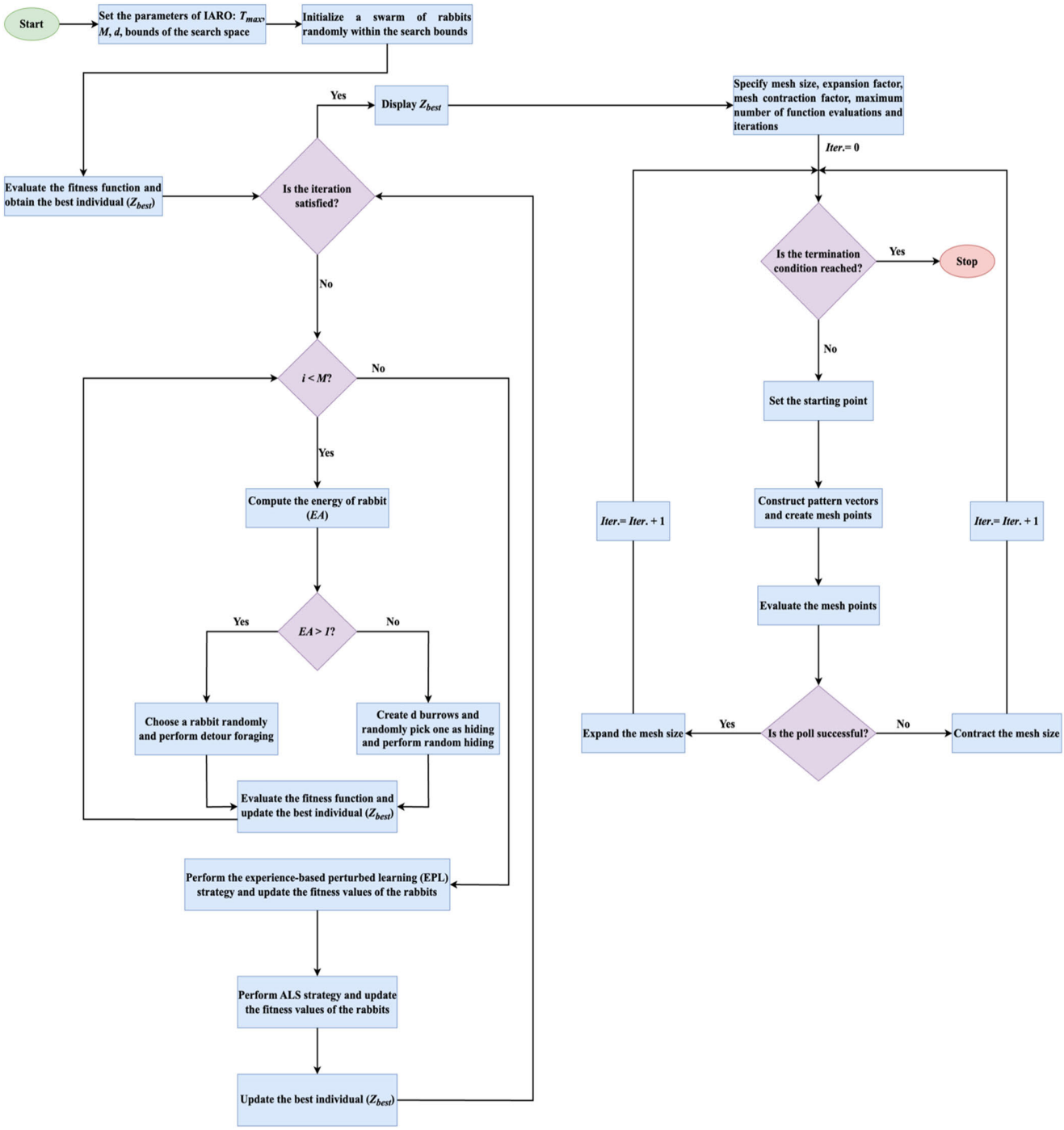

FIGURE 4. Flowchart for the proposed IARO-PS.

() function has been extended by the authors to assign tasks to VMs. The Space-shared policy is the basis for assigning tasks to VMs in the simulation. To reduce the average of the QoS management settings, independent jobs are picked for the randomized processing. The average value of 10 executions is the outcome that we have obtained. A 3-month workload dataset is considered to assess the performance of IARO-PS under real-world workload conditions. The Ames Research Centre, NASA, uploaded this dataset from 128 iPSC/860 servers [43]. 42,240 jobs with their completion times are included in the dataset; each work is regarded as a task. The simulation's population consists of ten potential solutions, with a maximum iteration of 100. Two experimental configurations are used to run divergent simulations to examine the range of resources available in the cloud. The outcomes are contrasted with those of other scheduling algorithms, such as

TABLE 4. The algorithmic computational settings for IARO-PS.

Maximum iterations	Search Space	Problem Dimensionality (D)	Population Size
100	[-100, 100]	50	50

TABLE 5. The algorithmic computational parameters for GA, QMPSO, MGWO and BSO.

Methodologies	Computational Parameters
GA	$\mu = 0.6, \eta = 0.02, r_1 = r_2 = 0.5$
QMPSO	$\omega = 1, C_1 = C_2 = 1.49, r_1 = r_2 = r_3 = 0.5,$ $\gamma = 0.5, CC_{high} = 0.8, CC_{low} = 0.35$
MGWO	$r_1 = r_2 = 0.5$
BSO	$C = S = 1.5, \ rand = rand_1 = rand_2 = rand_a = rand_b = 0.5, FL = 0.5$

MGWO [42], GA, BSO [48], and QMPSO [23]. Table 7 lists the specifications for technical modelling that were employed in simulation modelling in CloudSim.

1) EXPERIMENTAL SETUP 1: SIMULATION RESULTS WITH HOMOGENEOUS VMS (THE NUMBER OF VMS ARE KEPT CONSTANT AND THE NUMBER OF TASKS VARY)

Between 500 and 3000 autonomous, dynamic workloads are taken into consideration to complete the simulations. The 600 uniform virtual machines have been set up to schedule these jobs. Every simulation is run 10 times autonomously, and the average outcome is noted. For the duration of the simulation, the time span among two consecutive implementations is kept at 300*ms*.

Maintaining load balance is crucial for cutting down on makespan by keeping loads in balance. The suggested method combines the IARO-PS scheduling methods with load balancing. Evidently, in Figure 5 (a), the suggested technique allows for load trade-offs, which reduces the makespan, particularly in situations where there are more activities. Employing an efficient load balancing mechanism, the suggested method may be able to shorten the makespan. The workload collection is represented on the X-axis whereas the makespan is displayed on the Y-axis. Because there are more control parameters in MGWO [42] than in BSO and GA, it performs better for makespan. Despite being a useful method for striking a balance between exploration and exploitation, MGWO has produced noteworthy outcomes. Compared to QMPSO [23], the hybrid GAYA technique that has been suggested performs better for makespan reduction since QMPSO focuses more on task scheduling than load balancing. The suggested approach performs better than all other algorithms described because it can shorten response times, as seen in Figure 5 (b). Figure 5 (b) clearly shows that the overloaded state causes the latency (response time) to be higher at the early stages for a couple of workloads than it is for the duration with fewer workloads. The IARO-PS algorithm responds with a significantly shorter reaction time because of the minimization of makespan and appropriate usage of VM. The number of workloads is shown in the X-axis, whereas the response time is shown in the Y-axis. It is crucial to fully use resources when load balancing is done correctly. Because the suggested methodology distributes the loads among the VMs in an even manner using a load balancing strategy, it significantly improves VM utilization. Figure 5 (c) shows a graphic representation of the simulation result of VM utilization for different jobs specified. The ability to distribute loads uniformly makes the suggested strategy superior to previous approaches in terms of resource utilization. Effective equilibrium has been reached by the IARO-PS in keeping the loads. The reduction in makespan improves the DoI and keeps the system's loads consistent. As a result, the DoI is significantly reduced, as seen in Figure 5 (d). Simulation results indicate that the IARO-PS algorithm outperforms the previously mentioned algorithms by improving usage of resources by 45.23%, 36.74%, 19.75%, 14.31%, and reducing makespan by 15.35%, 10.45%, 4.35%, and 2.31%, respectively when compared with GA, QMPSO [23], MGWO [42], and BSO [48].

2) EXPERIMENTAL SETUP 2: SIMULATION RESULTS WITH HETEROGENEOUS VMS (NUMBER OF VMS DOES VARY, BUT THE NUMBER OF TASKS ARE CONSTANT)

This simulation has between 50 and 300 distinct kinds of VMs. 750 jobs are run through 10 independent executions, and the average results of these autonomous implementations are noted. For all of the simulations, the time delay between two consecutive implementations is kept constant at 300 *ms*.

The minimizing of the makespan for the various VMs is shown in Figure 6 (a). The makespan is shown in the Y-axis whereas the number of VMs is displayed on the X-axis. When there are 300 VMs, the performance of the QMPSO [23] and the IARO-PS is nearly the same. The IARO-PS technique outperforms the others in makespan reduction as the VM size grows with the jobs. When compared to the other methods given, the figure demonstrates a noteworthy performance. The suggested method performs better for reaction time than the other examined methods, as shown in Figure 6 (b). The number of VMs is represented on the X-axis while the reaction time is shown on the Y-axis. Because load balancing is taken into account and makespan is minimized, the suggested IARO-PS technique has better response times. As previously noted, Figure 6 (c) displays the experimental evaluation of resource utilization combined with additional techniques with heterogeneous VMs. The total number of VMs is displayed on the X-axis, while the amount of VMs utilized is shown on the Y-axis. The decrease in makespan is the reason for the success of increasing VM usage. The fitness function in our suggested strategy is taken into consideration in part because of the rise in VM utilization.

According to the simulation results, when compared to the aforementioned algorithms, the IARO-PS performs significantly better. Resource utilization is improved by 19.39%, 12.17%, 9.10%, and 6.02%, and makespan is reduced by 7.33%, 4.17%, 1.44%, and 1.03% when compared to BSO [48], GA, MGWO [42], and QMPSO [23], respectively.

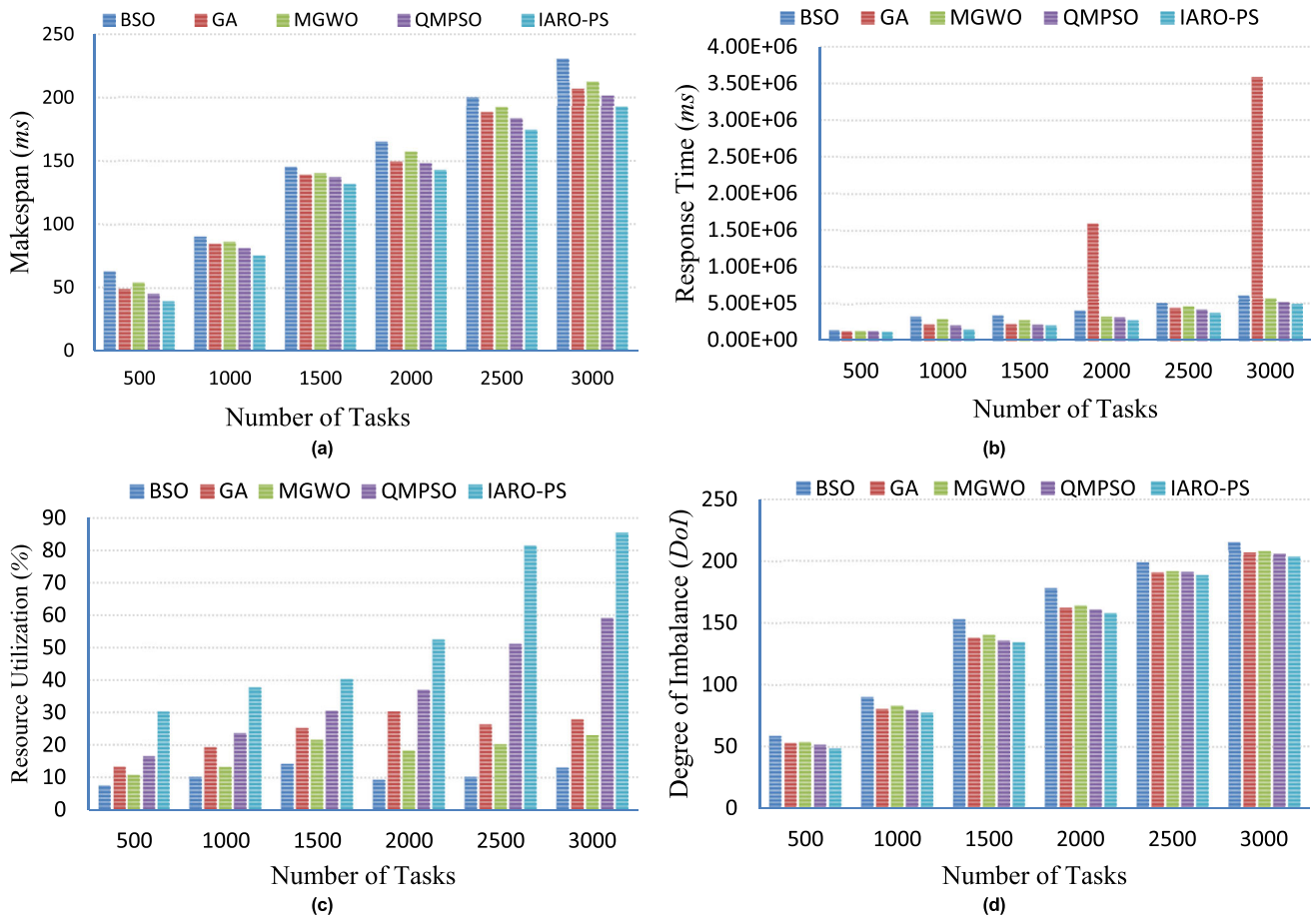

FIGURE 5. Simulation results for varying tasks (a) Makespan, (b) Response Time, (c) Resource or VM utilization, and (d) Degree of Imbalance.

When there are fewer jobs and VMs, there is no need for task scheduling or load balancing because there won't be any overloaded scenarios. When more jobs need to be scheduled across a variety of VMs, an issue occurs.

C. STATISTICAL ANALYSIS

Statistical analysis has been used to determine the IARO-PS's statistical significance, subjected to overall effectiveness. The performance of examined algorithms is compared using the Friedman test to identify any significant differences [44]. The outcomes of simulations are used to calculate Friedman's average ranking for each algorithm. To calculate the statistical significance, it is assigned to determine how one method differs from the others. Appendix Table 7 displays the algorithms' respective Friedman average rankings. To determine the statistical difference between the studied algorithms, the following null hypothesis (NH_0) is considered:

$\boldsymbol{NH_0}$: Every scheduling algorithm operates similarly.

Four algorithms and three performance parameters are taken into consideration for assessing the statistical significance. From 1 to k(k being the collection of techniques), the ranks are assigned, based on the values acquired for each algorithm's performance criteria. For example, the algorithm with the lowest average value (mean) is rated 1, and the method with the highest average value is ranked 4. When two or more algorithms provide the same result, the rank is determined by taking into account the average rank for each algorithm. For each technique, the ranks are shown inside the bracket in Table 7. The test's α is configured with a confidence level of 0.10. Eq. (43) is used to determine the Friedman average rank R_i of the i^{th} method.

$$R_i = \frac{Sum\ of\ total\ ranks\ obtained\ by\ the\ i^{th}\ algorithm}{Total\ number\ of\ performance\ parameters} \tag{43}$$

The statistics provided by the Friedman test are indicated in Eq. (44).

$$F_F = \frac{(P-1)X_F^2}{P\,(k-1) - X_F^2} \tag{44}$$

where $X_F^2 = \frac{12P}{k(k+1)}\left[\sum_{i=1}^{4} R_i^2 - \frac{k(k+1)^2}{4}\right]$, k being the collection of methodologies andthe collection of performance parameters is P.

Depending on the DoF, or the F-distribution with $(k-1)(P-1)$ and $(k-1)$, the Friedman statistics (F_F) are

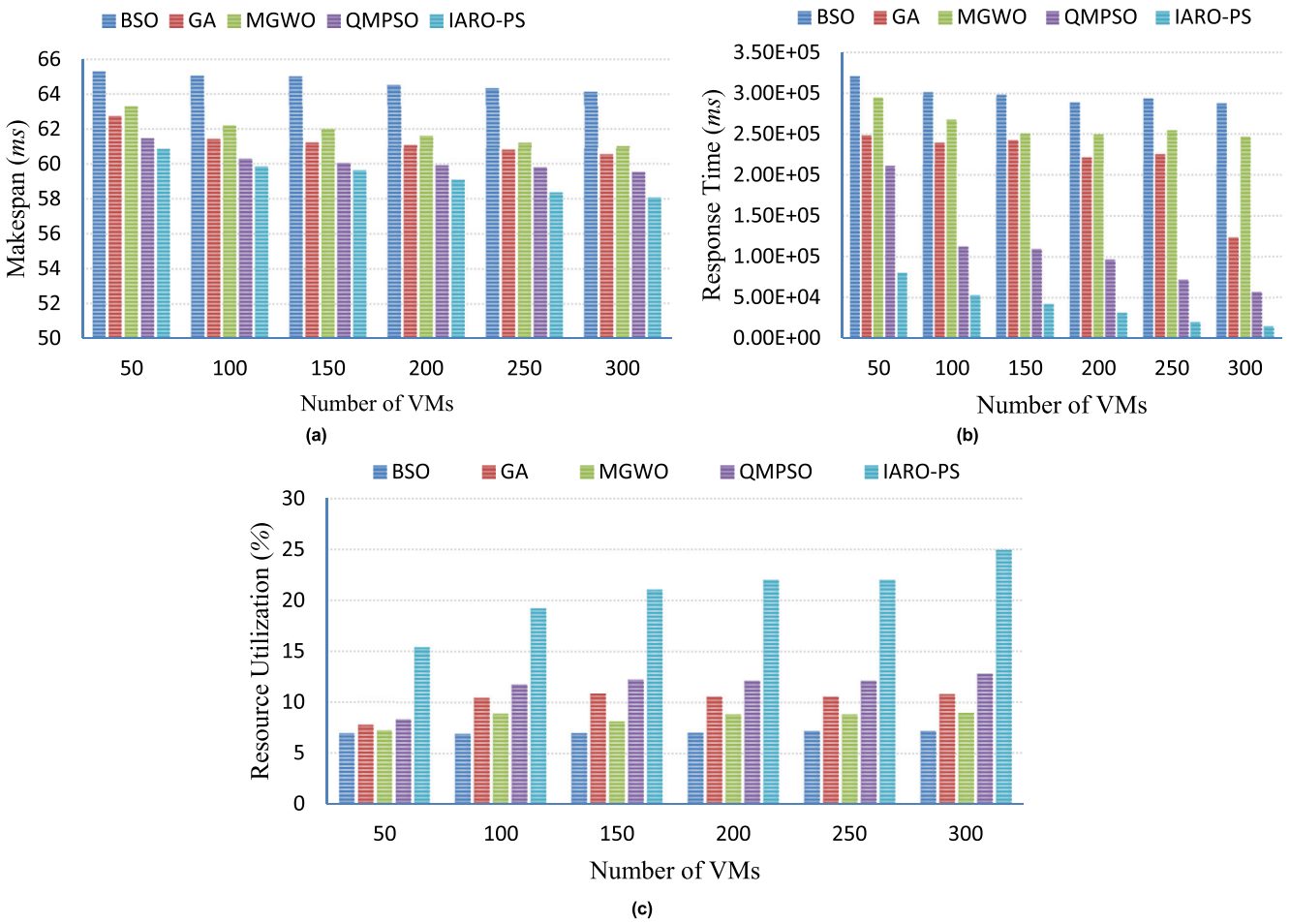

FIGURE 6. Simulation results for varying VMs (a) Makespan, (b) Response Time, and (c) Resource or VM utilization.

computed. For 4 methods and 3 performance factors, the estimated DoF ranges from 3 to 6. Consequently, for $\alpha = 0.10$, the critical estimation of $F(3, 6)$ is 5.28473 [45]. When the value of F_F exceeds the critical value, the null hypothesis NH_0 is discarded. The F_F estimation in our instance is 7.42, exceeding the criticalestimation of 5.28473. Consequently, the null hypothesis is disproved. It is evident from the Friedman statistical evaluation that the IARO-PS outperforms the other scheduling algorithms in terms of statistics. All other scheduling algorithms do not perform statistically identically.

VI. RESULT'S ANALYSIS

The efficacy of the suggested IARO-PS is assessed using a collection of 30 standard baseline functions, and the algorithms' statistical significance is ascertained using the Friedman test in MATLAB. Unimodal functions make up the first three and are utilized to determine the suggested method's exploitation capabilities based on the simulation outcomes shown in Table 6 (*referAppendix*). The suggested technique outperforms all previous algorithms for these three functions. Benchmark functions 4 through 10 have numerous local optima and are multimodal. These are used to determine the hybrid metaheuristic method's exploration capabilities. With the exception of F8 and F9, the proposed procedure yields a decent result. Benchmark functions 11 through 20 combine multimodal and unimodal elements. With the exception of F11, F14, F17, and F19, the suggested approach yields intermediate outcomes for these benchmark functions. Lastly, functions 21 through 30 in the row's baseline are composition functions that are employed to determine how much the hybrid method's capacity for exploration and exploitation is traded off. With the exception of F30, the applied IARO-PS approach performs noticeably better than the examined methods. Applying Friedman's average rank test, Friedman's rank is derived from the standard baseline functions. Table 7 (*refer Appendix*) illustrates the better value of the suggested approach over other compared methods.

Finding the optimal mapping for the tasks via a hybrid technique is the main contribution of this research. A superior PS with adjustable control parameters may also result in increased performance. Since these two factors are the main concerns of task scheduling in the issue space, a target function that is dependent on makespan and VM usage is considered in this work. Given these diverse jobs and VMs, the method might converge quite slowly. The loads are evenly distributed among the VMs using the suggested load

TABLE 6. Results in comparison of IARO-PS algorithm with GA, MGWO, and QMPSO in MATLAB.

Benchmark Functions	Algorithms	Best	Mean	Worst	Standard Deviation	Benchmark Functions	Best	Mean	Worst	Standard Deviation
F1	GA	1.82E+08	4.09E+08	7.17E+08	1.59E+08	**F16**	1.61E+03	1.61E+03	1.63E+03	3.67E-01
	MGWO	1.15E+09	1.65E+09	2.35E+09	4.03E+09		1.62E+03	1.62E+03	1.62E+03	3.74E-01
	QMPSO	3.72E+07	4.15E+07	5.00E+07	3.77E+07		7.70E+05	2.16E+06	3.78E+06	8.40E+05
	IARO-PS	1.73E+07	3.24E+07	6.37E+07	1.52E+07		3.64E+05	8.92E+05	1.48E+06	3.73E+05
F2	GA	1.55E+02	3.65E+02	3.44E+02	1.47E+02	**F17**	1.21E+07	4.75E+07	9.53E+09	2.35E+07
	MGWO	1.02E+11	1.31E+11	1.68E+11	1.96E+10		1.06E+08	2.24E+08	3.09E+08	7.04E+07
	QMPSO	1.52E+08	2.10E+08	2.60E+08	3.60E+07		1.63E+03	1.60E+08	1.60E+08	7.08E-01
	IARO-PS	4.45E+07	7.12E+07	1.24E+08	2.89E+07		1.66E+03	1.66E+03	1.66E+03	5.40E-01
F3	GA	5.59E+04	8.35E+04	1.22E+05	2.17E+04	**F18**	8.90E+07	2.10E+09	5.10E+09	1.63E+09
	MGWO	2.64E+05	3.66E+05	4.50E+05	5.54E+04		4.22E+09	9.42E+09	1.52E+10	3.81E+09
	QMPSO	4.79E+04	7.51E+04	9.59E+04	1.43E+04		6.18E+06	7.20E+06	9.26E+06	1.05E+06
	IARO-PS	1.43E+04	2.54E+04	3.41E+04	6.10E+03		1.35E+05	5.30E+06	8.71E+06	2.90E+06
F4	GA	1.82E+03	2.52E+03	3.30E+03	5.03E+02	**F19**	2.06E+03	2.11E+03	2.22E+03	5.94E+01
	MGWO	1.70E+04	2.30E+04	3.14E+04	4.69E+03		2.35E+03	2.68E+03	2.98E+03	2.49E+02
	QMPSO	4.63E+02	5.20E+02	6.38E+02	1.03E+02		1.95E+03	1.95E+03	1.94E+03	2.14E+00
	IARO-PS	4.72E+02	4.67E+02	4.89E+02	5.79E+02		1.94E+03	1.94E+03	1.94E+03	2.41E+00
F5	GA	5.19E+02	5.22E+02	5.22E+02	3.05E-02	**F20**	2.28E+04	8.01E+04	1.20E+05	3.19E+04
	MGWO	5.20E+02	5.20E+02	5.22E+02	3.42E-02		2.95E+05	6.95E+05	1.71E+06	4.22E+05
	QMPSO	5.19E+02	5.21E+02	5.21E+02	4.43E-02		1.37E+04	2.54E+04	5.61E+04	1.28E+04
	IARO-PS	5.20E+02	5.20E+02	5.20E+02	4.42E-02		4.71E+03	1.34E+04	2.16E+04	5.84E+03
F6	GA	6.50E+02	6.58E+02	6.63E+02	4.85E+00	**F21**	1.44E+04	6.76E+06	1.49E+07	4.08E+00
	MGWO	6.60E+02	6.66E+02	6.70E+02	3.68E+00		2.95E+05	4.36E+07	7.50E+07	1.68E+07
	QMPSO	6.50E+02	6.55E+02	6.65E+02	4.63E+00		1.37E+04	2.23E+05	2.84E+05	4.79E+04
	IARO-PS	6.48E+02	6.52E+02	6.60E+02	3.49E+00		4.71E+03	7.04E+04	3.93E+05	6.89E+04
F7	GA	8.61E+02	9.39E+02	1.04E+03	5.02E+01	**F22**	2.60E+03	3.15E+03	3.60E+03	3.13E+02
	MGWO	1.92E+03	2.14E+03	2.46E+03	1.69E+02		4.41E+03	6.61E+03	2.16E+04	5.28E+03
	QMPSO	6.99E+02	7.01E+02	7.04E+02	4.59E-01		2.45E+03	2.80E+03	3.21E+03	2.27E+02
	IARO-PS	7.01E+02	7.01E+02	7.03E+02	1.25E-01		2.43E+03	2.78E+03	3.30E+03	2.42E+02
F8	GA	1.02E+03	1.05+03	1.10E+03	2.77E+01	**F23**	2.78E+03	2.83E+03	2.88E+03	2.97E+01
	MGWO	1.46E+03	1.51E+03	1.55E+03	2.55E+01		3.74E+03	4.24E+03	4.85E+03	3.28E+02
	QMPSO	1.12E+03	1.21E+03	1.34E+03	6.51E+01		2.66E+03	2.67E+03	2.68E+03	8.86E+00
	IARO-PS	1.06E+03	1.19E+03	1.31E+03	6.89E+01		2.69E+03	2.69E+03	2.70E+03	1.10E+01
F9	GA	1.30E+03	1.35E+03	1.49E+03	5.91E+03	**F24**	2.66E+03	2.70E+03	2.73E+03	2.13E+01
	MGWO	1.68E+03	1.74E+03	1.82E+03	4.26E+01		2.76E+03	2.87E+03	3.04E+03	8.66E+01
	QMPSO	1.44E+03	1.57E+03	1.65E+03	7.53E+01		2.58E+00	2.58E+03	2.58E+03	1.10E-01
	IARO-PS	1.26E+03	1.38E+03	1.46E+03	6.79E+01		2.67E+00	2.60E+03	2.61E+03	7.43E-02
F10	GA	5.22E+03	6.91E+03	8.10E+03	8.40E+02	**F25**	2.81E+03	2.83E+03	2.85E+03	1.32E+01
	MGWO	1.29E+04	1.45E+04	1.63E+04	9.86E+02		2.79E+03	2.86E+03	2.94E+03	4.65E+01
	QMPSO	8.21E+03	9.60E+03	1.65E+04	6.00E+02		2.70E+03	2.70E+03	2.71E+03	2.07E+00
	IARO-PS	5.56E+03	6.21E+03	7.63E+03	5.14E+02		2.72E+03	2.71E+03	2.71E+03	1.35E+00
F11	GA	8.35E+03	8.62E+03	1.01E+04	2.05E+02	**F26**	2.72E+03	2.72E+03	2.72E+03	5.98E-01
	MGWO	1.38E+04	1.43E+03	1.53E+04	4.89E+02		2.71E+03	2.71E+03	2.71E+03	6.15E-01
	QMPSO	7.56E+03	8.66E+03	1.19E+04	1.25E+03		2.75E+03	2.75E+03	2.75E+03	4.36E-02
	IARO-PS	7.71E+03	8.76E+03	1.04E+04	8.92E+02		2.70E+03	2.70E+03	2.70E+03	5.90E-02

TABLE 6. *(Continued.)* **Results in comparison of IARO-PS algorithm with GA, MGWO, and QMPSO in MATLAB.**

F12	GA	1.23E+03	1.22E+03	1.23E+03	2.81E-01	**F27**	4.37E+03	4.56E+03	4.77E+03	2.53E+02
	MGWO	1.20E+03	1.20E+03	1.20E+03	3.34E-01		4.67E+03	4.75E+03	4.85E+03	5.93E+01
	QMPSO	1.20E+03	1.20E+03	1.20E+03	4.21E-01		3.14E+03	3.13E+03	3.14E+03	4.90E+00
	IARO-PS	1.20E+03	1.20E+03	1.20E+03	3.26E-01		3.11E+03	3.13E+03	3.15E+03	1.30E+01
F13	GA	1.32E+03	1.34E+03	1.34E+03	4.52E-01	**F28**	3.89E+03	4.57E+03	5.00E+03	3.25E+02
	MGWO	1.31E+03	1.31E+03	1.31E+03	6.38E-01		4.10E+03	4.69E+03	5.31E+03	3.78E+02
	QMPSO	1.33E+03	1.33E+03	1.33E+03	4.58E-02		3.25E+03	3.32E+03	3.34E+03	2.90E+01
	IARO-PS	1.30E+03	1.30E+03	1.30E+03	5.41E-02		3.23E+03	3.27E+03	3.27E+03	1.32E+01
F14	GA	1.45E+03	1.50E+03	1.53E+03	2.01E+01	**F29**	2.25E+06	4.33E+07	7.92E+07	2.16E+07
	MGWO	1.59E+03	1.63E+03	1.66E+03	2.17E+01		4.04E+07	5.96E+07	8.55E+07	1.37E+07
	QMPSO	1.40E+03	1.41E+03	1.42E+03	2.62E-02		3.17E+03	1.37E+04	8.00E+04	2.39E+04
	IARO-PS	1.43E+03	1.43E+03	1.43E+03	4.25E-02		3.14E+03	4.75E+03	1.08E+04	2.63E+03
F15	GA	2.92E+04	2.05E+05	4.22E+05	1.16E+03	**F30**	1.29E+05	5.52E+05	7.12E+06	1.99E+05
	MGWO	3.20E+07	6.52E+07	1.09E+08	2.75E+07		5.54E+00	1.96E+06	3.49E+06	1.07E+06
	QMPSO	1.53E+03	1.55E+03	1.55E+03	3.95E+00		4.04E+03	4.72E+03	5.70E+03	4.69E+02
	IARO-PS	1.55E+03	1.56E+03	1.56E+03	3.65E+00		4.33E+03	4.84E+03	5.78E+03	4.60E+02

balancing mechanism. Subsequently, the suggested IARO-PS method is used to dynamically execute the individual jobs in the cloud environment. Two distinct experimental setups are used for the experiments: (1) with homogeneous resources by altering the assigned workloads, and (2) using diverse resources while upholding the assigned tasks. In all cases, the IARO-PS algorithm significantly improves the taken-into-account QoS scheduling metrics.

To handle work scheduling for the cloud environment, authors in [23] have merged the altered PSO with the Q-learning technique. Compared with the present technique, it is less efficient because the load alone has been taken into account as a component of the target function. Not taken into account are makespan and resource utilization, which are the main goals of the load balancing challenge. Because the load balancing method hasn't been put into practice, the resources haven't been fully utilized, which prevents it from producing the best results.

The MGWO algorithm was employed by the researchers in [42] to schedule tasks. The variety of requests with heterogeneous resources has not been considered in this research. For the aforementioned constraint, the present strategy performs better than this approach due to the lack of makespan and resource utilization. Because their approach takes into account a homogeneous environment, it is inefficient while handling dynamic independent requests. Similarly, the presence of multiple conflicting control factors renders the BSO [48] inferior to the existing methodology. Furthermore, these control factors help the convergence to happen slowly.

Unlike earlier works, the current approach takes into account incompatible scheduling requirements such as response time, DoI, makespan, and resource utilization to develop a load balancing strategy. The efficiency of the current technique stems from its consideration of task compatibility with the VMs on which the jobs are to be processed. The IARO algorithm does better at convergently reaching the global optima since it is designed to balance exploration and exploitation. The current method is scalable and generic due to the PS's versatility.

VII. CONCLUSION AND FUTURE INSIGHTS

In cloud computing, creating a task scheduling algorithm that works well is crucial for improving resource utilization and cutting down on makespan by maintaining system equilibrium. Being precise,load balancing and task scheduling amongst VMs are the primary topics of this study. It focuses on scheduling the workloads and uniformly mapping those to the VMs by combining two metaheuristic methodologies. The current article offers two contributions. To establish a balance between the VMs' loads in the initial phase, an efficient load balancing policy is used. The jobs' compatibility for the relocation and the resources that are being underutilized are planned to determine which VMs receive which tasks. The ideal task-resource mapping is called upon in the second fold by PS, and it is subsequently supplied as input to IARO. After that, PS schedules the work. By balancing local and worldwide search, this hybrid approach greatly enhanced the exploring capability. A baseline function is used to accomplish the hybridization procedure to assess its effectiveness. Results from the experiments over MGWO, QMPSO, and GA is satisfactory. Additionally, CloudSim uses a mix of baseline datasets of NASA and biological samples to schedule dynamically separate jobs. As mentioned in the previous section, two distinct settings were used for the simulation. Based on the simulations, the IARO-PS algorithm performed better than the previously mentioned

TABLE 7. Friedman rank of the proposed IARO-PS algorithm with GA, MGWO, and QMPSO obtained from benchmark functions of CEC2017.

Benchmark Functions	GA	MGWO	QMPSO	IARO-PS
F1	4.09E+08 (3)	1.65E+09 (4)	4.15E+07 (2)	**3.24E+07 (1)**
F2	3.65E+02 (1)	1.31E+11 (4)	2.10E+08 (3)	7.12E+07 (2)
F3	8.36E+04 (3)	3.66E+05 (4)	7.51E+04 (2)	**2.54E+04 (1)**
F4	2.52E+03 (3)	2.30E+04 (4)	5.20E+02 (2)	**4.76E+02 (1)**
F5	5.22E+02 (4)	5.22E+02 (2)	5.22E+02 (2)	5.22E+02 (2)
F6	6.57E+02 (3)	6.65E+02 (4)	6.55E+02 (2)	**6.52E+02 (1)**
F7	9.39E+02 (3)	2.14E+03 (4)	7.00E+02 (1.5)	7.00E+02 (1.5)
F8	1.05+03 (1)	1.51E+03 (4)	1.22E+03 (3)	1.20E+03 (2)
F9	1.35E+03 (1)	1.74E+03 (4)	1.56E+03 (3)	1.39E+03 (2)
F10	6.92E+03 (2)	1.44E+04 (4)	9.60E+03 (3)	**6.21E+03 (1)**
F11	8.62E+03 (2)	1.44E+03 (1)	8.66E+03 (3)	8.77E+03 (4)
F12	1.23E+03 (4)	1.21E+03 (2)	1.21E+03 (2)	1.21E+03 (2)
F13	1.33E+03 (3.5)	1.32E+03 (2)	1.33E+03 (3.5)	**1.31E+03 (1)**
F14	1.50E+03 (3)	1.63E+03 (4)	1.41E+03 (1)	1.43E+03 (2)
F15	2.05E+05 (3)	6.52E+07 (4)	1.55E+03 (1)	1.57E+03 (2)
F16	1.63E+03 (1.5)	1.63E+03 (1.5)	2.16E+06 (4)	8.92E+05 (3)
F17	4.75E+07 (2)	2.24E+08 (4)	1.60E+08 (3)	**1.66E+03 (1)**
F18	2.10E+09 (3)	9.42E+09 (4)	7.20E+06 (2)	**5.30E+06 (1)**
F19	2.12E+03 (3)	2.69E+03 (4)	1.95E+03 (2)	**1.93E+03 (1)**
F20	8.00E+04 (3)	6.94E+05 (4)	2.54E+04 (2)	**1.34E+04 (1)**
F21	6.75E+06 (3)	4.36E+07 (4)	2.22E+05 (2)	**7.03E+04 (1)**
F22	3.15E+03 (3)	6.61E+03 (4)	2.80E+03 (2)	**2.78E+03 (1)**
F23	2.83E+03 (3)	4.24E+03 (4)	2.68E+03 (1)	2.69E+03 (2)
F24	2.70E+03 (3)	2.87E+03 (4)	2.58E+03 (1)	2.60E+03 (2)
F25	2.83E+03 (3)	2.87E+03 (4)	2.71E+03 (1)	2.72E+03 (2)
F26	2.73E+03 (3)	2.72E+03 (2)	2.76E+03 (4)	**2.70E+03 (1)**
F27	4.57E+03 (3)	4.75E+03 (4)	3.13E+03 (1.5)	3.13E+03 (1.5)
F28	4.57E+03 (3)	4.69E+03 (4)	3.32E+03 (2)	**3.26E+03 (1)**
F29	4.32E+07 (3)	5.96E+07 (4)	1.37E+04 (2)	**4.75E+03 (1)**
F30	5.52E+05 (3)	1.96E+06 (4)	4.72E+03 (1)	4.83E+03 (2)
Average Rank (Ar)	**2.63**	**3.55**	**2.15**	**1.57**

algorithms, reducing makespan by 10.45% (GA), 2.31% (QMPSO), 4.35% (MGWO), 15.35% (BSO), and 4.17% (GA), 1.03% (QMPSO), 1.44% (MGWO), 7.33% (BSO), in both homogeneous and heterogeneoussurroundings, respectively, and improving resource utilization by 36.74% (GA), 14.31% (QMPSO), 19.75% (MGWO), 45.23% (BSO) and 12.17% (GA), 6.02% (QMPSO), 9.10% (MGWO), 19.39% (BSO).

In the coming years, approaches considering ML algorithms could also be applied to minimize a datacenter's power usage and estimate the processing capability of its computer resources. To assess the variety of scheduling techniques, other quality of service characteristics like execution time, energy usage, cost, and migration time could be included.

CONFLICT OF INTEREST

The authors of this research paper confirm that they have no conflict of interest to disclose.

APPENDIX

Simulation results of the IARO-PS algorithm in MATLAB and Friedman Rankings are shown in Table 6 and Table 7.

REFERENCES

[1] M. Kaushik and M. Santosh, "A state-of-art on cloud load balancing algorithms," *Int. J. Comput. Digit. Syst.*, vol. 9, no. 2, pp. 201–220, Jan. 2020.

[2] A. Mahapatra, K. Mishra, R. Pradhan, and S. K. Majhi, "Next generation task offloading techniques in evolving computing paradigms: Comparative analysis, current challenges, and future research perspectives," *Arch. Comput. Methods Eng.*, vol. 31, no. 3, pp. 1405–1474, Apr. 2024, doi: 10.1007/s11831-023-10021-2.

[3] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4) pp. 50–58, 2010.

[4] P. Mell and T. Grance, "The NIST definition of cloud computing," *Nat. Inst. Sci. Technol.*, vol. 800, p. 145, 2011.

[5] A. Mahapatra, K. Mishra, S. K. Majhi, and R. Pradhan, "EFog-IoT: Harnessing power consumption in fog-assisted of things," in *Proc. IEEE Region Symp. (TENSYMP)*, Jul. 2022, pp. 1–6, doi: 10.1109/TENSYMP54529.2022.9864457.

[6] M. Xu, W. Tian, and R. Buyya, "A survey on load balancing algorithms for virtual machines placement in cloud computing," *Concurrency Comput., Pract. Exper.*, vol. 29, no. 12, p. e4123, Jun. 2017.

[7] J. O. Gutierrez-Garcia and A. Ramirez-Nafarrate, "Agent-based load balancing in cloud data centers," *Cluster Comput.*, vol. 18, no. 3, pp. 1041–1062, Sep. 2015.

[8] A. Mahapatra, K. Mishra, S. K. Majhi, and R. Pradhan, "Latency-aware Internet of Things scheduling in heterogeneous fog-cloud paradigm," in *Proc. 3rd Int. Conf. Emerg. Technol. (INCET)*, May 2022, pp. 1–7, doi: 10.1109/INCET54531.2022.9824613.

[9] J. D. Ullman, "NP-complete scheduling problems," *J. Comput. Syst. Sci.*, vol. 10, no. 3, pp. 384–393, Jun. 1975.

[10] O. H. Ibarra and C. E. Kim, "Heuristic algorithms for scheduling independent tasks on nonidentical processors," *J. ACM*, vol. 24, no. 2, pp. 280–289, Apr. 1977.

[11] M. Kalra and S. Singh, "A review of metaheuristic scheduling techniques in cloud computing," *Egyptian Informat. J.*, vol. 16, no. 3, pp. 275–295, Nov. 2015.

[12] S. A. Murad, Z. R. M. Azmi, A. J. M. Muzahid, M. K. B. Bhuiyan, M. Saib, N. Rahimi, N. J. Prottasha, and A. K. Bairagi, "SG-PBFS: Shortest gap-priority based fair scheduling technique for job scheduling in cloud environment," *Future Gener. Comput. Syst.*, vol. 150, pp. 232–242, Jan. 2024.

[13] P. Banerjee, S. Roy, A. Sinha, M. M. Hassan, S. Burje, A. Agrawal, A. K. Bairagi, S. Alshathri, and W. El-Shafai, "MTD-DHJS: Makespan-optimized task scheduling algorithm for cloud computing with dynamic computational time prediction," *IEEE Access*, vol. 11, pp. 105578–105618, 2023, doi: 10.1109/ACCESS.2023.3318553.

[14] S. A. Murad, Z. R. M. Azmi, A. J. M. Muzahid, M. M. H. Sarker, M. S. U. Miah, M. K. B. Bhuiyan, N. Rahimi, and A. K. Bairagi, "Priority based job scheduling technique that utilizes gaps to increase the efficiency of job distribution in cloud computing," *Sustain. Comput., Informat. Syst.*, vol. 41, Jan. 2024, Art. no. 100942, doi: 10.1016/j.suscom.2023.100942.

[15] K. Sharma, M. M. Hassan, S. K. Pandey, M. Saini, R. Doriya, M. K. Ojha, A. Sinha, C. Kaushal, A. K. Bairagi, and N. F. Soliman, "Cloud based multi-robot task scheduling using PMW algorithm," *IEEE Access*, vol. 11, pp. 146003–146013, 2023, doi: 10.1109/ACCESS.2023.3344459.

[16] K.-M. Cho, P.-W. Tsai, C.-W. Tsai, and C.-S. Yang, "A hybrid meta-heuristic algorithm for VM scheduling with load balancing in cloud computing," *Neural Comput. Appl.*, vol. 26, no. 6, pp. 1297–1309, Aug. 2015.

[17] M. Shojafar, S. Javanmardi, S. Abolfazli, and N. Cordeschi, "FUGE: A joint meta-heuristic approach to cloud job scheduling algorithm using fuzzy theory and a genetic method," *Cluster Comput.*, vol. 18, no. 2, pp. 829–844, Jun. 2015.

[18] S. Dam, G. Mandal, K. Dasgupta, and P. Dutta, "Genetic algorithm and gravitational emulation based hybrid load balancing strategy in cloud computing," in *Proc. 3rd Int. Conf. Comput., Commun., Control Inf. Technol. (CIT)*, Feb. 2015, pp. 1–7, doi: 10.1109/C3IT.2015.7060176.

[19] V. Jeyakrishnan and P. Sengottuvelan, "A hybrid strategy for resource allocation and load balancing in virtualized data centers using BSO algorithms," *Wireless Pers. Commun.*, vol. 94, no. 4, pp. 2363–2375, Jun. 2017.

[20] X. J. Wei, W. Bei, and L. Jun, "SAMPGA task scheduling algorithm in cloud computing," in *Proc. 36th Chin. Control Conf. (CCC)*, Jul. 2017, pp. 5633–5637, doi: 10.23919/ChiCC.2017.8028252.

[21] H. B. Alla, S. B. Alla, and A. Ezzati, "A priority based task scheduling in cloud computing using a hybrid MCDM model," in *Proc. 3rd Int. Symp.*, Casablanca, Morocco, May 2017, pp. 235–246, doi: 10.1007/978-3-319-68179-5_21.

[22] S. Rani and P. K. Suri, "An efficient and scalable hybrid task scheduling approach for cloud environment," *Int. J. Inf. Technol.*, vol. 12, no. 4, pp. 1451–1457, Dec. 2020.

[23] U. K. Jena, P. K. Das, and M. R. Kabat, "Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 34, no. 6, pp. 2332–2342, Jun. 2022.

[24] A. Mahapatra, S. K. Majhi, K. Mishra, R. Pradhan, D. C. Rao, and S. K. Panda, "An energy-aware task offloading and load balancing for latency-sensitive IoT applications in the fog-cloud continuum," *IEEE Access*, vol. 12, pp. 14334–14349, 2024.

[25] F. Ebadifard, S. M. Babamir, and S. Barani, "A dynamic task scheduling algorithm improved by load balancing in cloud computing," in *Proc. 6th Int. Conf. Web Res. (ICWR)*, Apr. 2020, pp. 177–183, doi: 10.1109/ICWR49608.2020.9122287.

[26] R. Khorsand, M. Ghobaei-Arani, and M. Ramezanpour, "A self-learning fuzzy approach for proactive resource provisioning in cloud environment," *Softw., Pract. Exper.*, vol. 49, no. 11, pp. 1618–1642, Nov. 2019.

[27] E. Rafieyan, R. Khorsand, and M. Ramezanpour, "An adaptive scheduling approach based on integrated best-worst and VIKOR for cloud computing," *Comput. Ind. Eng.*, vol. 140, Feb. 2020, Art. no. 106272.

[28] S. T. Milan, L. Rajabion, H. Ranjbar, and N. J. Navimipour, "Nature inspired meta-heuristic algorithms for solving the load-balancing problem in cloud environments," *Comput. Oper. Res.*, vol. 110, pp. 159–187, Oct. 2019.

[29] M. Abdullahi, M. A. Ngadi, and S. M. Abdulhamid, "Symbiotic organism search optimization based task scheduling in cloud computing environment," *Future Gener. Comput. Syst.*, vol. 56, pp. 640–650, Mar. 2016.

[30] V. Polepally and K. Shahu Chatrapati, "Dragonfly optimization and constraint measure-based load balancing in cloud computing," *Cluster Comput.*, vol. 22, no. 1, pp. 1099–1111, Jan. 2019.

[31] R. Nanduri, N. Maheshwari, A. Reddyraja, and V. Varma, "Job aware scheduling algorithm for MapReduce framework," in *Proc. IEEE 3rd Int. Conf. Cloud Comput. Technol. Sci.*, Nov. 2011, pp. 724–729, doi: 10.1109/CLOUDCOM.2011.112.

[32] F. Ebadifard and S. M. Babamir, "A PSO-based task scheduling algorithm improved using a load-balancing technique for the cloud computing environment," *Concurrency Comput., Pract. Exper.*, vol. 30, no. 12, p. e4368, Jun. 2018.

[33] M. Sommer, M. Klink, S. Tomforde, and J. Hähner, "Predictive load balancing in cloud computing environments based on ensemble forecasting," in *Proc. IEEE Int. Conf. Autonomic Comput. (ICAC)*, Jul. 2016, pp. 300–307, doi: 10.1109/ICAC.2016.16.

[34] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable cloud computing environments and the CloudSim toolkit: Challenges and opportunities," in *Proc. Int. Conf. High Perform. Comput. Simul.*, Jun. 2009, pp. 1–11, doi: 10.1109/HPCSIM.2009.5192685.

[35] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw., Pract. Exper.*, vol. 41, no. 1, pp. 23–50, Jan. 2011.

[36] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis With Applications to Biology, Control, and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1992.

[37] R. Venkata Rao, "Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems," *Int. J. Ind. Eng. Computations*, vol. 7, no. 1, pp. 19–34, 2016.

[38] H. M. Pandey, "Jaya a novel optimization algorithm: What, how and why?" in *Proc. 6th Int. Conf. - Cloud Syst. Big Data Eng.*, Jan. 2016, pp. 728–730, doi: 10.1109/CONFLUENCE.2016.7508215.

[39] K. Mishra, J. Pati, and S. Kumar Majhi, "A dynamic load scheduling in IaaS cloud using binary Jaya algorithm," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 34, no. 8, pp. 4914–4930, Sep. 2022.

[40] T. Prakash, V. P. Singh, S. Singh, and S. Mohanty, "Binary Jaya algorithm based optimal placement of phasor measurement units for power system observability," *Energy Convers. Manag.*, vol. 140, pp. 34–35, Jan. 2017.

[41] M. Zhang, H. Ren, and C. Xia, "A dynamic placement policy of virtual machine based on MOGA in cloud environment," in *Proc. IEEE Int. Symp. Parallel Distrib. Process. Appl. IEEE Int. Conf. Ubiquitous Comput. Commun. (ISPA/IUCC)*, Dec. 2017, pp. 885–891, doi: 10.1109/ISPA/IUCC.2017.00135.

[42] F. A. Saif, R. Latip, Z. M. Hanapi, and K. Shafinah, "Multi-objective grey wolf optimizer algorithm for task scheduling in cloud-fog computing," *IEEE Access*, vol. 11, pp. 20635–20646, 2023.

[43] D. G. Feitelson and B. Nitzberg, "Job characteristics of a production parallel scientific workload on the NASA Ames iPSC/860," in *Proc. Workshop Job Scheduling Strategies Parallel Process.*, Berlin, Germany, Apr. 1995, pp. 337–360.
[44] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Dec. 2006.
[45] (2018). *F Distribution Table*. [Online]. Available: http://www.socr.ucla.edu/Applets.dir/F_Table.html
[46] T. Wang, Z. Liu, Y. Chen, Y. Xu, and X. Dai, "Load balancing task scheduling based on genetic algorithm in cloud computing," in *Proc. IEEE 12th Int. Conf. Dependable, Autonomic Secure Comput.*, Aug. 2014, pp. 146–152, doi: 10.1109/DASC.2014.35.
[47] K. Mishra, R. Pradhan, and S. K. Majhi, "Quantum-inspired binary chaotic salp swarm algorithm (QBCSSA)-based dynamic task scheduling for multiprocessor cloud computing systems," *J. Supercomput.*, vol. 77, no. 9, pp. 10377–10423, Sep. 2021.
[48] K. Mishra and S. K. Majhi, "A binary bird swarm optimization based load balancing algorithm for cloud computing environment," *Open Comput. Sci.*, vol. 11, no. 1, pp. 146–160, Jan. 2021.
[49] L. Wang, Q. Cao, Z. Zhang, S. Mirjalili, and W. Zhao, "Artificial rabbits optimization: A new bio-inspired meta-heuristic algorithm for solving engineering optimization problems," *Eng. Appl. Artif. Intell.*, vol. 114, Sep. 2022, Art. no. 105082.
[50] D. Khamari, R. K. Sahu, T. S. Gorripotu, and S. Panda, "Automatic generation control of power system in deregulated environment using hybrid TLBO and pattern search technique," *Ain Shams Eng. J.*, vol. 11, no. 3, pp. 553–573, Sep. 2020.

SANTOSH KUMAR PAUL received the M.C.A. and M.Tech. (C.Sc.) degrees. He is currently pursuing the Ph.D. degree in computer science with Sri Sri University, Cuttack. He has 15 years of experience in teaching. He is also working in load balancing optimization algorithms in cloud computing.

SUNIL KUMAR DHAL received the M.Tech. and Ph.D. degrees in computer science from Utkal University, Bhubaneswar, Odisha. He is currently a Professor in information system with the Faculty of Management Studies, Sri Sri University, Cuttack, Odisha. He has supervised five Ph.D. scholars. He has 25 years of teaching experience. He has many publications and patents. He has authored many of the books. His current research interests include quantitative methods for management, software project management, data analysis, management science, cloud computing, wireless sensor networks, and ERP.

SANTOSH KUMAR MAJHI (Senior Member, IEEE) received the B.Tech. degree from the Veer Surendra Sai University of Technology (VSSUT), Burla, Sambalpur, India, the M.E. degree in CSE from the PG Department of Computer Science and Application, Utkal University, Bhubaneswar, and the Ph.D. degree in computer and information systems from Sri Sri University, Cuttack, India. He is currently an Associate Professor with the Department of Computer Science and Information Technology, Guru Ghasidas Viswavidyalaya (Central University), Bilaspur, Chhattisgarh, India. He has published more than 70 numbers of papers in various journals of repute and conference proceedings. His research interests include cloud computing, security analytics, data mining, and AI. He was a recipient of the Young Scientist Award from VIFRA, India, the Young IT Professional Award from the Computer Society of India, and the Best Faculty Award from VSSUT.

ABHIJEET MAHAPATRA received the B.Tech. degree in computer science and engineering (CSE) from BPUT, Rourkela, Odisha, India, and the M.Tech. degree in CSE from the Veer Surendra Sai University of Technology (VSSUT), Burla, Sambalpur, India, where he is currently pursuing the Ph.D. degree in CSE. He has published three journals and two conference papers. His research interests include task scheduling and load balancing with IoT management in cloud and fog computing.

PRADOSH KUMAR GANTAYAT received the M.Tech. degree from KIIT University, Odisha, and the Ph.D. degree from the VSS University of Technology, Odisha, India. He is currently an Assistant Professor with The ICFAI Foundation for Higher Education, Hyderabad, specializes in computer science and engineering. His research interests include security issues in computer networks, soft computing, AI, and nature-inspired algorithms. He boasts a prolific publication record with 25 papers in esteemed international journals, conferences, and book chapters. He received the prestigious "Research Excellence Award 2021" from InSc, Bengaluru, and holds three Indian patents to his credit. A member of LMISTE and LMIAENG, he is recognized for his expertise and contributions to the field.

SWARUPA PANDA received the B.Tech. and M.Tech. degrees in CSE from the Veer Surendra Sai University of Technology (VSSUT), Burla, Sambalpur, India. She is currently an Assistant Professor on Contract with the Department of Computer Science and Engineering, Government College of Engineering, Keonjhar, Odisha.