Received 3 March 2023, accepted 3 April 2023, date of publication 11 April 2023, date of current version 19 April 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3266294

RESEARCH ARTICLE

A Multi-Objective Based Scheduling Framework for Effective Resource Utilization in Cloud Computing

PILLAREDDY VAMSHEEDHAR REDDY AND KARRI GANESH REDDY, (Member, IEEE)
School of Computer Science and Engineering, VIT-AP University, Amaravathi 522237, India
Corresponding author: Karri Ganesh Reddy (guncity11@gmail.com)

ABSTRACT Cloud computing is a promising platform for running massive workflow applications based on a pay-per-use model. In cloud computing, the reduction of energy consumption and providing security to workflow scheduling are the key research areas. The primary focus of the existing algorithms, viz., particle swarm optimization (PSO), crow Search optimization (CSO) and other non-metaheuristic algorithms like Round Robbin (RR), SJF, Min-Min, Min-Max etc., is based on the execution time and cost of the workflow applications as a budget constraint. However, these algorithms failed to adequately determine energy consumption, resource utilization, and security in workflow scheduling. To address this issue, a multi-objective scheduling framework is proposed. In this paper, the framework performs dynamic workflow scheduling using universal unique identification- Blake (UUID-Blake), Manhattan Distance-Partition around algorithm (MD-PAM), Linear Scaling-Crow Search Optimization (LS-CSO), Anova-Recurrent Neural Network. The implementation of this framework was achieved in three phases (Phase 1, Phase 2, and Phase 3). Phase 1 is about user registration and authentication using UUID-Blake, which enhances security by allowing legitimate users into the cloud environment. Phase 2 deals with clustering and resource monitoring using MD-PAM and A-RNN, to reduce makespan the similar tasks are clustered using task length and maximize the resource utilization by predicting the resource availability. Phase 3 deals with the scheduling of dynamic workflows using LS-CSO by selecting suitable virtual machines. We have considered the heterogeneous computing scheduling problem (HCSP) and grid workload archive (GWA)-T-12 Bitbrains datasets for comparing our proposed framework with existing works. Based on the result analysis, the proposed LS-SCO outperformed when compared with the algorithms CSO, PSO and RR has achieved better performance.

INDEX TERMS Crow swarm optimization, linear scaling, Manhattan distance, partitioning around medoid, recurrent neural network, workflow scheduling.

I. INTRODUCTION

Cloud computing is transforming into a high-performance computing environment with enormous computational resources such as storage, networking, databases, and applications. These resources are allocated to users from the collective resource pool with reduced management or interaction [1]. The three major services delivered by the cloud technology include Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), along with Infrastructure-as-a-Service (IaaS) [2] out of which the most prominent modelis IaaS. This model provides pre-configured virtual machines (VMs) to the user from the cloud framework [3], [4]. An infinite quantity of computational resources and storage facilities can be attained by using VMs [5].With the increased demand for such cloud services, the number of users is expanding day by day [6]. Because of this, the unpredictable resource usage of VMs has increased, due to which the servers of cloud data centers get either over-utilized or under-utilized, thereby resulting in an imbalanced state [7], [8]. As a result,

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

it degrades the total performance and usage of resources among the cloud servers. To utilize resources for maximum potential and to accomplish the tasks in minimum time, the development of efficient workflow algorithms is required [9].

The prominent approach in modelling high data processing applications is the workflow, which is performed in cloud computing (CC) domains [10]. The Direct Acyclic Graph (DAG) denotes the workflow where the computational works are illustrated by graph nodes and dependencies among the graph's tasks are illustrated by graph edges [11]. The scientific application has a huge impact on the DAG size [12]. If the scientific application is simple together less complicated, the workflow size is tiny, or else it is taken as large [13]. The methodology, which performs workflow tasks mapping on the heterogeneous together with distributed resources of a computing system is termed Workflow Scheduling (WFS). To satisfy the user-defined constraints, a suitable quantity of resources is allotted for the workflow tasks execution [14].

A. SCIENTIFICAPPLICATIONS

A set of application criteria must be identified in or- der to better model scientific applications, identify the appropriate leased resources for execution, and define a better scheduling strategy, such as:

1) Parallel model: specifies whether the application is defined as a single sequential task or as multiple parallel tasks that may be executed on multiprocessor machines or distributed processing nodes.
2) Task dependency: relating to the I/O flow between tasks; indicating that a task cannot begin execution until the output data of studies on which it depends are available.
3) Resource Usage: it specifies the applications are I/O – Intensive, Data-Intensive and CPU-Intensive.

The majority of scientific applications have a highly dependent task structure that is intended to be executed in a distributed computing environment. This form of a collection of tasks in these applications refers to scientific workflows. Figure 1-5 shows a collection of popular scientific workflows that simulate real- world scientific applications. Cybershake Figure.1 and Broad Band Figure.2 for earthquake science, Montage Figure.3 for astronomy research, LIGO Figure.5 for gravitational wave detection, and Epigenomics Figure.4 for biological concerns. Each scientific workflow type will be resolved by carrying out a specific set of tasks, as denoted by colors in Fig. 1, depending on the related application. Additionally, a single task or several parallel tasks may be used to complete each task. As a result, a collection of tasks arranged in a specific dependency structure and grouped into a set of particular jobs produces the scientific workflows.

CPU intensive foundation and Memory-bound base applications require more physical memory for executing the tasks, and I/O-intensive workflows consist of tasks that consume and generate huge quantities of data, and hence spend the majority of their time executing I/O operations.

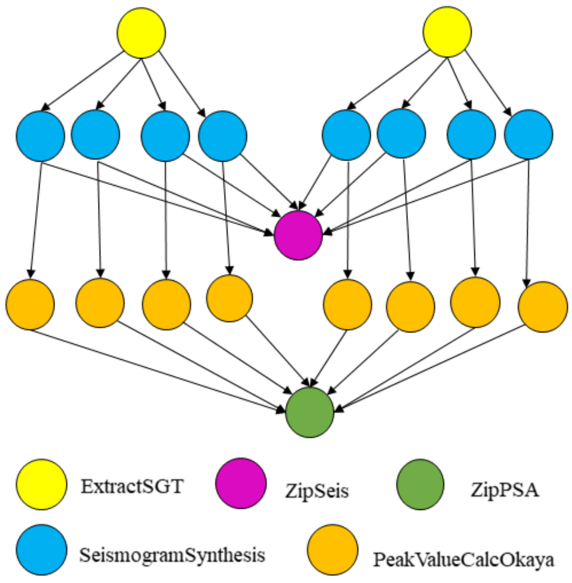

FIGURE 1. Cybershake.

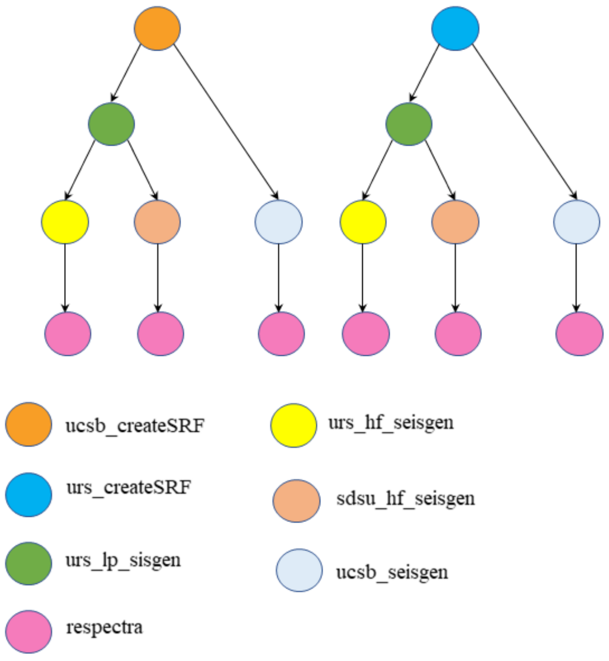

FIGURE 2. Broadband.

In workflow scheduling, various workflow models are there, viz., i) Workflow in parallel, ii) fork-join workflow, and iii) random workflow.

Workflows are used in many scientific fields to describe complex computational problems that can be solved efficiently in a distributed environment. Cyber Shake, Epigenomics, LIGO, broad bond, and montage workflow are some of the example workflows. Several methodologies have been proposed for elevating the efficacy of the WFS process.

Although the basic principles of cloud computing (CC)such as elasticity and the heterogeneity of cloud resources are not considered in the existing works [15].

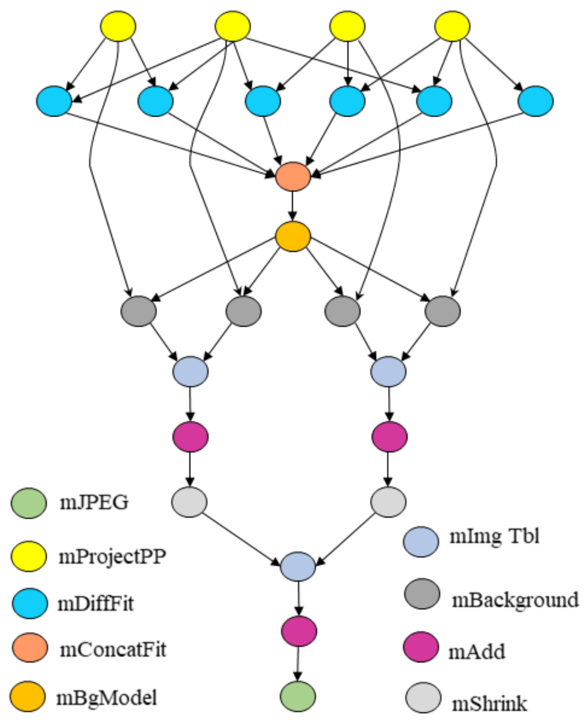

FIGURE 3. Montage.

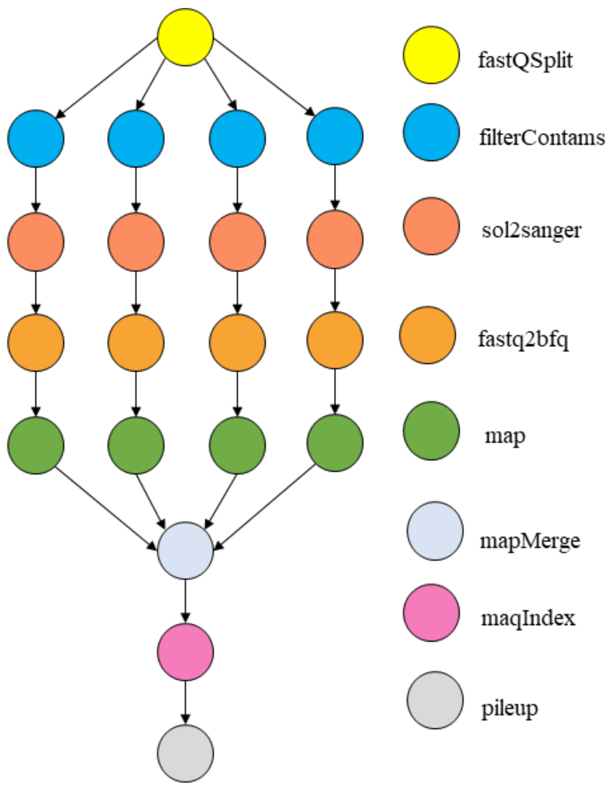

FIGURE 4. Montage.

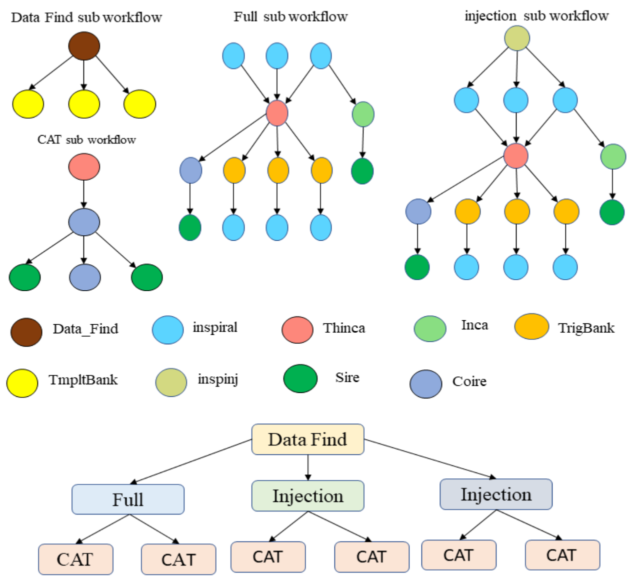

FIGURE 5. LIGO.

A focus on a single workflow is made on most of the scheduling procedures in cloud environments and the issue of inconsistent arrivals in large-scale WFS is ignored [16]. Additionally, some important constraints such as energy consumption (EC),Resource utilization, and Security are not considered by most works [17]. Securely hiring the tasks to suitable machines with reduced EC is a crucial task in the distributed data-intensive computing environment. The existing works failed to solve these problems [18].

In this paper, we develop a framework, which addresses the energy constraint, under/over utilization of resources and security constraints to schedule the dynamic workloads. In our proposed framework, we integrate the LS-CSO and ANOVA-RNN with the UUID-BLAKE hashing technique to resolve the above-mentioned issues.

Contribution: The key highlights of this paper is as follows.

i. We proposed a resource-efficient framework based on workflow scheduling for cloud computing using UUID-BLAKE, MH-PAM, A-RNN, and LS-CSO.
ii. We developed a secure workflow scheduling policy by generating hash codes. The hash code is generated by using the UUID-BLAKE algorithm to improve the authenticity of the cloud server by allowing the legitimate user to access it.
iii. To reduce the makespan and system overhead, similar tasks are clustered. The clustering process is done with the help of MH-PAM. PAM is a k-medoid algorithm, which has less sensitive noise and outliers compared to k-means.
iv. The virtual machines are monitored by using A-RNN, which predicts the usage of the resources. Based on the availability of resources, a dynamic workflow is generated. To overcome the drawback of RNN, the ANOVA radial basis kernel function is incorporated.
v. The most appropriate virtual machines to perform the dynamic workflow are selected by means of the LS-CSO

TABLE 1. Abbreviations.

Word	Meaning
CSO	CrowSearchoptimization
PSO	ParticleSwarmOptimization
LS-CSO	LinearScaling-CrowSearchoptimization
MD-PAM	ManhattanDistance-partitionaroundmedoid
A-RNN	Annona-RecurrentNeuralNetwork
UUID	UniversalUniqueIdentification
DAG	DirectAcyclicGraph
CC	CloudComputing
WFS	Workflowscheduling
WMS	WorkflowManagementSystem
VM	VirtualMachine
HL	HiddenLayer
AP	AwarenessProbability
TS	TaskScheduling
MS	MakeSpan
ET	ExecutionTime

algorithm. Each VM's capability is evaluated, and tasks are then assigned to the most appropriate one. By doing so, energy consumption is reduced and resources are utilized efficiently.

The remaining sections of the paper are organized as follows:

The outline of the paper is condensed as follows: Section II surveys the associated works regarding the proposed method. Section III explains the proposed methodology called a resource-efficient and secure workflow scheduling. Section IV illustrates the results and discussion of the proposed method based on performance metrics. Finally, Section V concludes the paper with future work and Table.1 represents the abbreviations used in the paper.

II. RELATED WORK

Dubey et al. [19] framed a management system for the evaluation of multi-organizations in a community cloud model. An Ideal Distribution Approach (IDA) in conjunction with an Enhanced IDA (EIDA) heuristic-cantered algorithm was demonstrated for resource allocation. These algorithms identified the attainable and ideal schedule in workflow execution and reduced the Makespan (MS) and cost during the time limit efficiently. In the need to meet SLA terms and to reduce time and computation costs, the IDA algorithm attained solutions efficiently. The IDA algorithm's efficacy was elevated by load balance phase inclusion. While analogizing with the IDA algorithm, Enhanced IDA (EIDA) attained a reduced MS. The experimental outcomes illustrated that EIDA performed superior regarding cost and MS than the existing algorithms. However, resource utilization was not executed efficiently.

For data-intensive scientific workflows, Hazekampet al. [20] combined Static and Dynamic Storage Management. the three-tiered approach was formulated: (1) to analyze the storage requirements of a workflow before execution, a static analysis algorithm contributed an authentic prediction for success or failure; (2) Ignoring the deadlock that occurred at runtime, an online storage management algorithm was supposed to be for the storage requirements of future tasks. (3) The storage consumption of distinctive tasks was limited by an online storage management algorithm that allowed strong guarantees of the static evaluation along with dynamic management algorithms. As the outcomes suggest, the execution of these methods on three complex workflows performed superior to the existing techniques. Nevertheless, elevated workflow MS was demanded in this scheme.

In Asghari et al. [21], they introduced a framework that consisted of numerous cooperative agents. Here, Task Scheduling (TS) phases along with resource provisions were taken into account, and the QoS imparted to the consumer is confined. All TS along with resource provisioning processes were contributed by the integrated model, and it also served as a management tool for user applications and efficacious exploitation of cloud resources. It had a complex scheduling process for the reason that it relied on sub-tasks, and it performed well on dependent simultaneous tasks. Regarding MS, resource utilization, and cost, together with consumption of energy, high-quality performance was attained in the outcomes when analogized with other cloud resource management models. However, the impact of communication among tasks has not been inspected.

Kim et al. [22] presented human-intelligence workflow management (HIWM) for the dynamic resource distribution, storage, work processing, and the computation of operations for fast Augmented Reality (AR) service provision on diverse smart mobile devices that relied on human attitude and was applicable for future web environments. Relying on the description of the metadata along with AR user requests, pre-processing was executed to reduce service response time in HIWM. A dynamic job distribution model was demonstrated for the processing of big data for AR services. It also comprises the cloud infrastructure depending on the computer's capability. With regards to outcomes grounded on AR service requests, 40.56% of operation time was reduced when compared to the existing models, whereas inefficiency occurs in the process of a complex workflow.

Patnaiket al. [23] elucidated a workflow-cantered technique for TS in a cloud environment. A huge quantity of co-dependent tasks by the cluster of computed resources with heterogeneous capability was performed by this workflow procedure. Here, efficient allotment of tasks was the vital motive, as a consequence of which MS was reduced. Additionally, to indicate a set of interdependent tasks, an elongation of the max-min algorithm was executed on a DAG.

TABLE 2. Workflow evaluations and Objectives used in Algorithms.

Algorithm	Type of Workflow Used						Scheduling Objectives							
	Montage	SIPHT	LIGO	Cybershake	Epigenomics	Random workflow	Makespan	Resource Utilization	Deadline	Cost	Reliability	Security	Energy Consumption	Budget
BTS [26]	✓	✓	✓	✓	✓	✓	-	-	✓	✓	-	-	-	-
EMO WFS in cloud [8]	✓	✓	✓	✓	✓	-	✓	-	-	✓	-	-	-	-
NMMWS [27]	✓	-	-	-	-	-	✓	✓	-	-	-	-	-	-
HMOPSO for WFS [28]	✓	✓	✓	✓	✓	-	-	-	-	✓	-	-	✓	-
RSO [29]	-	-	-	-	✓	-	✓	✓	-	✓	-	-	-	-
HMO Workflow Scheduling in IaaS Cloud [2]	✓	✓	✓	✓	-	-	✓	-	-	-	-	-	-	-
GRP-HEFT[30]	✓	✓	-	✓	-	-	✓	✓	-	✓	-	-		✓
Task Replication[31]	✓	✓	✓	✓	-	-	-	-	-	✓	✓	-	-	-
BDCWS[6]	✓	-	✓	-	✓	-	-	-	✓	✓	-	-	-	✓
MOWS[9]	✓	-	-	✓	-	-	✓	-	-	-	-	✓	-	-
TOPSIS[32]	✓	-	✓	-	✓	-	-	✓	✓	✓	-	-	-	-
ALO-PSO[33]	✓	-	-	-	-	-	✓	-	-	✓	✓	✓	-	-
GMPSO[34]	✓	✓	-	✓	✓	-	✓	✓	-	-	-	-	-	-
REEWS[35]	-	-	-	-	-	✓	-	-	-	-	✓	-	✓	-
CPCS[36]	✓	✓	-	✓	-	✓	-	-	✓	-	✓	✓	-	-

Regarding MS, this technique outperformed the existing methods when testing was executed on standard scientific workflows. While executing the workflow process, ignorance of some basic principles of CC was the downside here.

Belgacem and Beghdad-Bey [24] proposed HEFT-ACO trade-off between cost and makespan is designed to improve the quality of service requirements. The authors are primarily concerned about finding the best solution for assigning tasks to the VM. For assigning tasks the workflow schedule is divided into two parts, the first of which uses HEFT to schedule tasks based on rank and the second of which uses ACO to schedule ready tasks. The experimental results showed that HEFT-ACO outperformed the current algorithms in terms of cost and MS. However, energy consumption and security were not carried out efficiently.

Haidri et al. [25] proposed a cost-effective deadline-aware algorithm (CEDA) to optimize the execution cost and time by satisfying deadline constraints. CEDA chooses the tasks with the highest rank and sends them to the cheapest VMs, taking the VMs' acquisition delay into account. The experiment results showed that CEDA outperformed the ICICP and ICICPD2 in terms of cost and time. To reduce the cost, instead of creating a new VM instance, CEDA prefers to use an existing one whose remaining charge period is long enough to complete the task before its latest finish time. However, the utilization of resources and energy consumption is efficient.

The preceding Table 2 provides an overview of workflow evaluations and scheduling objectives used in various algorithms. It also reveals that the authors considered the montage, SIPHT, and LIGO workflows as static with limited parameters, resulting in poor scheduling performance. The author BTS considered all workflows, including random workflows, but did not take into account multi-objective parameters such as makespan, resource utilization, and energy consumption. The existing scheduling algorithms did not address secure scheduling; security is a major issue in cloud computing in terms of unauthorized persons accessing and modifying data (tasks), which can lead to a different scheduling attack. To address the aforementioned limitations, our proposed approach addressed secure scheduling in the beginning by allowing authorized users into the cloud environment using the UUID-BLAKE hashing technique, and it addressed VM monitoring using A-RNN to improve scheduling performance in terms of resource utilization and energy consumption.

III. PROPOSED FRAMEWORK

In a cloud-computing paradigm, workflow scheduling is a key issue and poses a continuously challenging problem. Large numbers of virtual machines are employed in the cloud infrastructure. Thus, the selection of the most appropriate virtual machines for each task is a complex process. So, the work has proposed an resource-efficient and secure workflow scheduling in a cloud server using MH-PAM, LS-CSO, and ARNN with the UUID-BLAKE hashing technique. The proposed framework Figure.6 concentrates on three important phases. The first phase ensures the secure authorization of the users to the cloud server. In the second phase, the works requested by the users are scheduled to the server in such a way that the total time, as well as the energy required for the task's completion, is minimum. Then, the third phase focuses on the efficient selection of virtual machines that are appropriate for the user-requested task. By which the resource consumption, time consumption, as well as energy consumption can be drastically reduced.

A. PHASE 1: USER AUTHENTICATION

1) USER REGISTRATION

Initially, the users are registered with the corresponding cloud server by providing their details and their cloud server requirements such as bandwidth utilization, Hard disk drive, RAM, a processor with estimation details, and some other details.

2) GENERATION OF HASH CODE

Once the registration process has been completed, the hash code is generated for every registered user. This hash code generation process improves the authenticity of the cloud server by allowing the legitimate user to access the cloud server. In this work, the hash code is generated by using the UUID-BLAKE Hashing algorithm. Here, the universally unique identifier (UUID) is incorporated with the existing BLAKE hashing algorithm to improve the hash code complexity.

a: GENERATION OF HASH CODE USING UUID-BLAKE ALGORITHM

The BLAKE-32 hash function is used to hash a message ms and before that the message is padded with equal or more than 66bits,then it becomes to the multiple of 512.The representation of bits in message are consider as, the last 64 bits are binary representation of the bit length of the unpadded message. This padding message again splits into 512-bit blocks and reiteratively inputted to the compression function. With old, hash value, until now 64-bit counter bits hashed and optional 128-bit salt. Moreover, in the documentation the old one hash value is specified as initialization vector to the first block. While padding the counter last block is set to zero. In this work, the UUID is given as the input to the BLAKE algorithm to ensure the complexity of the hash code [37].

Following are the steps incorporated in the BLAKE algorithm.

The BLAKE's compressive function has '4' values as its input, which is expressed as,

- Chaining value $(H) = H_0, \ldots\ldots, H_7$

H (0——7) are the internal states of hash

- Message block (ms) $= ms_0, \ldots\ldots\ldots, ms_{15}$

The algorithm starts with splitting the user message ms into 512b blocks ms0 to ms15 (If necessary, the final block is null padded).

- Salt $(S) = S_0, \ldots\ldots, S_3$

The salt is chosen optionally by the user and set to a null value when no salt is required. Which is only used tasks, like randomized hashing.

- Counter $(C) = C_0$,C1

If the last block does not contain any bits from the original message, the counter is reset to zero.

The compression function of H, ms, S, C is

$$H' = \text{compress}(H, \text{ms}, S, C \tag{1}$$

Initialization, round iteration, and finalization are the '3' stages undergone by the compressive function.

Initialization:

The initial states of $L_0 \ldots\ldots\ldots L_{15}$is represented by 4×4 matrix.

$$L = \begin{pmatrix} L_0 & L_1 & L_2 & L_3 \\ L_4 & L_5 & L_6 & L_7 \\ L_8 & L_9 & L_{10} & L_{11} \\ L_{12} & L_{13} & L_{14} & L_{15} \end{pmatrix} \tag{2}$$

A 512-bit state L is maintained within the compress function, which is represented as a 4×4 matrix of 32-bit words. The current hash, salt value, timer value C, and a 256-bit constant c are used to initialize this state. The compression function's initial state is given

$$\begin{pmatrix} H_0 & H_1 & H_2 & H_3 \\ H_4 & H_5 & H_6 & H_7 \\ S_0 \oplus C_0 & S_1 \oplus C_1 & S_2 \oplus C_2 & S_3 \oplus C_3 \\ S_4 \oplus C_4 & S_5 \oplus C_5 & S_6 \oplus C_6 & S_7 \oplus C_7 \end{pmatrix} \tag{3}$$

The state matrix is iterated over in 10 rounds after being initialized. It is strongly recommended that simpler rounds be used when designing BLAKE, as this has been proven to increase security.

Round Function:

In every round it consists of 8 states those are, $\lambda_0 \ldots\ldots\ldots$ λ_7 that are responsible for changing the data (confusion) of

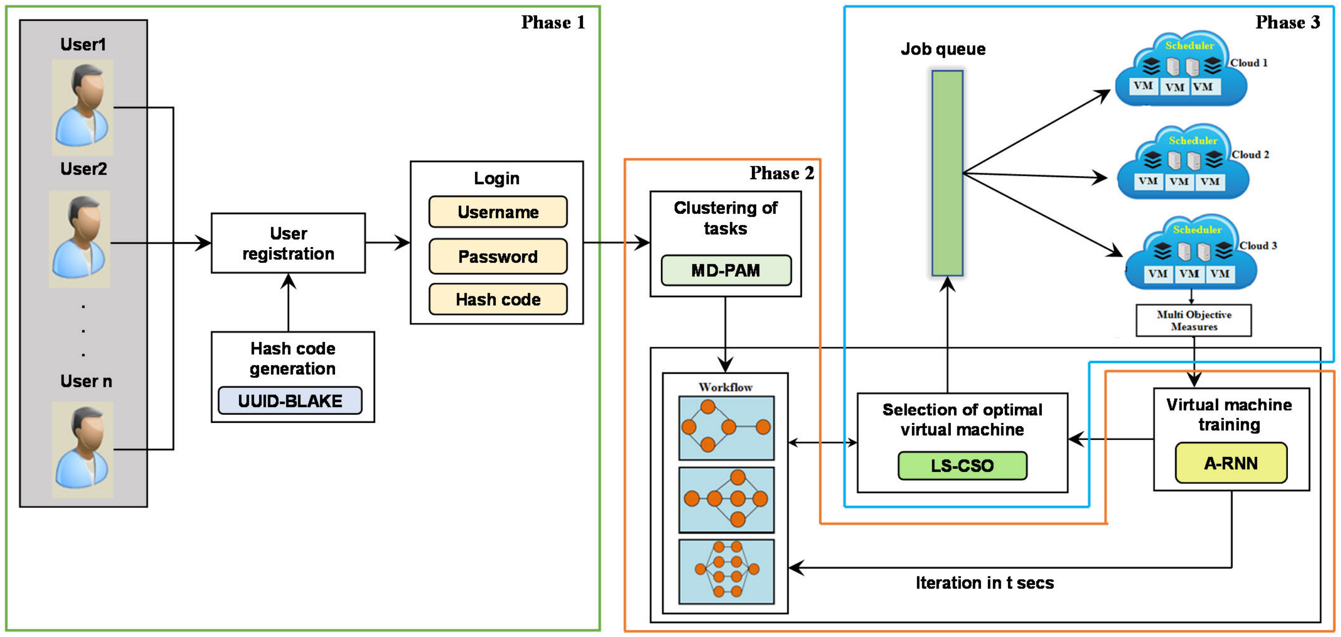

FIGURE 6. Multi-objective Framework using LS-CSO.

the BLAKE algorithm

$$\lambda_0(L_0, L_4, L_8, L_{12})\lambda_1(L_1, L_5, L_9, L_{13})$$
$$\lambda_2(L_2, L_6, L_{10}, L_{14})\lambda_3(L_3, L_7, L_{11}, L_{15})$$
$$\lambda_4(L_0, L_5, L_{10}, L_{15})\lambda_5(L_1, L_6, L_{11}, L_{12})$$
$$\lambda_6(L_2, L_7, L_8, L_{13})\lambda_7(L_3, L_4, L_9, L_{14})$$
$$a \leftarrow \mathrm{a+b+ms}_{\sigma \mathrm{Ru}(2\mathrm{i})} \oplus C\sigma_{\mathrm{Ru}}(2\mathrm{i}+1) \quad (4)$$
$$\mathrm{d} \leftarrow (\mathrm{d} \oplus \mathrm{a}) \ggg 16 \quad (5)$$
$$\mathrm{c} \leftarrow \mathrm{c+d} \quad (6)$$
$$\mathrm{b} \leftarrow (\mathrm{b} \oplus \mathrm{c}) \ggg 12 \quad (7)$$
$$\mathrm{a} \leftarrow \mathrm{a+b} + (\mathrm{ms}_{\sigma Ru(2i+1)} \oplus C_{\sigma Ru(2i)}) \quad (8)$$
$$\mathrm{d} \leftarrow (\mathrm{d} \oplus \mathrm{a}) \ggg 8 \quad (9)$$
$$\mathrm{c} \leftarrow \mathrm{c+ d} \quad (10)$$
$$\mathrm{b} \leftarrow (\mathrm{b} \oplus \mathrm{c}) \ggg 7 \quad (11)$$

Here, the constants are specified as C_j for $j = 0,\ldots\ldots\ldots\ldots,15$ and permutations of Z_{16} are signified as σ_{Ru} forRu = 0,1,.........,9.In the BLAKE documentation, both of them are proffered. Lastly, the output hash value H is formulated as.

Finalization:

A final step is carried out by the compressive function after ten/fourteen iterations of the λ transformation, The new chain values H are extracted from $L_0 \ldots\ldots\ldots\ldots L_{15}$ with salt and chain value.

$$H_i' \leftarrow H_i \oplus S_{imod4} \oplus H_i \oplus H_{i+8} fori = 0, 1 \ldots., 7 \quad (12)$$

After successfully generating the hash code, it is saved on both the cloud server and the user's system. When logging into the cloud, the user must authenticate with their username, password, and hash code.

3) LOGIN

After the successful generation of the hash code, the user must enter the username, password, and hash code. Then, these three data are verified by the verification server. If all entered details are correct, then the system allows the user to access the cloud server. Thus, the users can make use of cloud resources to complete their work.

B. PHASE 2: EFFICIENT WORKFLOW SCHEDULING

1) ATTRIBUTES EXTRACTION

After Submitting the tasks, the most important and required attributes such as the number of CPUs required for the task execution N_{CPU}, the number of instructionsN_{Ins}, and the dimensions such as large, medium and, small tasks D are extracted from the workflow. These extracted attributes highly contributed to the efficient clustering of tasks. The mathematical representation for the extracted attributes $WF_{Attributes}$ is given by.

$$WF_{Attributes} = \{N_{CPU}, N_{Ins}, D\}$$

2) TASK CLUSTERING

After that, the tasks are clustered based on the extracted attributes. This clustering process drastically reduces the system's overhead by assigning one or more small tasks into a single execution unit called a job. In this proposed framework, the clustering of similar tasks is done by the means of Manhattan Distance-based Partitioning around Medoid (MD-PAM). The PAM is a kind of K-Medoid Algorithm.

A k-medoid algorithm is a clustering approach that is related to the k-means clustering algorithm. Thus, the K-medoid-based PAM algorithm is less sensitive to noise and outliers compared to k-means because it uses medoids as cluster centers instead of means, which is used in k-means. In the PAM algorithm, the minimum distance between the medoids and the tasks is calculated in terms of Manhattan distance (MD), thereby the more similar tasks are formed as a cluster. Thus, the MD-PAM algorithm drastically improves the clustering accuracy.

Step 1: In an initial step, the optimal medoid k is initialized by adding the number of objects with minimum distance to all other objects. Here, the objects represent the tasks. The distances between the objects i and j at A_i and B_j respectively are calculated by the Manhattan distance formula that is given as.

$$MD = \sum_{i=1}^{n} |A_i\text{-}B_j| \tag{13}$$

Step 2: An object i (iϵO) is considered as a candidate that is added to k. Then, a total gainTG_i is computed for each object and that is given by.

$$TG_i = \sum\nolimits_{j\in 0} \text{Max}\{H_j - \partial(j,i),0\} \tag{14}$$

where, j is the object of O excepti. In case, $H_j > \eth(i,j)$ then the quality of the clustering is improved.

Step 3: After the calculation of the total gain of all objects in O, the object G, which has maximum,TG_g is selected and is mathematically formulated as.

$$K = K \cup \{G\} \tag{15}$$

$$O = O - \{G\} \tag{16}$$

These steps are repeated until the v objects are selected.

Step 4: The swap phase improves the clustering quality by optimizing the set of selected objects that are terminated by considering all swap pairs (v, $(\boldsymbol{v}, \boldsymbol{\emptyset})\ \boldsymbol{\epsilon k} \cup \boldsymbol{O}$ and computing the effect ξ_v on the sum of dissimilarities between objects and their cluster centers by swapping v and Φ, and then transferring Φ from O to k. $\xi_{v\Phi}$ is calculated by.

$$\xi_{v\phi} = \sum \left\{V_{tv\phi} | t \in O\right\} \tag{17}$$

where, $\boldsymbol{v_{tv\Phi}}$ denotes the contribution of each object t in O to swap v and O. If $\Phi(t,v) > H_t$ or $\Phi(t,v) = H_t$, then $\boldsymbol{v_{tv\Phi}}$ can be computed as.

$$v_{tvO} = \begin{cases} Min\{\partial(t,\phi) - H_t, 0\} & \partial(t,v) > H_t \\ Min\{\partial(t,\phi), \varepsilon_t\} - H_t & \partial(t,v) = H_t \end{cases} \tag{18}$$

Step 5: The pair (v,Φ) with the minimum $\xi_{k\Phi}$ is selected to determine whether the swapping is done or not. If $\xi k\Phi < 0$, then the swapping process is taken place and returns to the beginning phase of the swap. Otherwise, record the medoids.

Step 6: Then, the objects which are closest to the medoids and the minimum dissimilarities between the objects are grouped into the clusters and that is given by.

$$C_p = \{C_1, C_2, C_3, \ldots\ldots. C_n\} \tag{19}$$

where, $\{C_1, C_2, C_3, \ldots\ldots., C_n\}$ denotes the number of formed clusters. This cluster contains more similar tasks so that the tasks can be executed efficiently with limited consumption of resources and time.

3) WORKFLOW MANAGEMENT SYSTEMS

After the successful clustering, the number of requests is collected from the number of users and these requests are submitted to the workflow management systems (WMS). Usually, the heavy workload demands significant amounts of computing resources, at some point, the server exhausts its resources, and it fails to handle incoming requests. So, to deal with that, the WMS has been used. Thus, the WMS makes sure the efficient usage of cloud resources and manages the limited consumption of energy. The WMS resides on the host machine, which schedules the workflow tasks based on the available resource. Furthermore, it monitors the execution and manages the input data, intermediate files, and output files of the workflow tasks.

4) WORKFLOW MAPPER

The workflow mapper produces an executable workflow based on the availability of VMs, which are calculated (monitored and predicted), by ARNN. This workflow mapper identifies the suitable software along with hardware resources needed for the implementation. Thus, with limited ET, the number of tasks is executed. Furthermore, for performance optimization, the mapper restructures the workflow.

5) MULTI-OBJECTIVE MEASURES

From the number of virtual machines that reside on the cloud server the multi-objective measures such as network transmit throughput, memory usage, disk read throughput, disk write throughput, cpu cores, timestamp, cpu usage, memory capacity provisioned, and cpu capacity provisioned are extracted.

6) VIRTUAL MACHINE MONITORING

In this phase virtual machines are monitored, before assigning the workflows to the virtual machines are monitored based on the historical data of machines and extracted multi-objectives measures. Thus, the VMs are monitored using the A-RNN Figure.7The traditional RNN uses the sigmoid or softmax activation function for classification. However, the vanishing gradient that exists in the back-propagation process is the disadvantage of these activation functions, resulting in learning delay as well as poor classification performance. To overcome the aforementioned issues, the ANOVA radial basis kernel function, which is well-suited for deeper networks, is combined with the RNN algorithm. Furthermore, the error rate is reduced.

RNN is a neural network, In which the outcome from the previous step is used as input for the next step. The hidden state is the important feature of RNN, which retains the information of sequence. Each hidden layer (HL) in a traditional neural network has its own set of weights and biases. For example, the weights and biases of HL1, HL2 and HL3 are

(W1, B2), (W2, B2) and (W3, B3), respectively. It indicates that every HL is independent to the other. Thus, they do not remember the o/p of the previous layer. The RNN on the other hand, converts independent into dependent activation by assigning the same weights and biases to all the layers. Thus, by feeding each o/p into the next HL, the complexity of increasing the parameter and remembering each previous output is reduced.

Step 1: By iterating the following sequence from t = 1 to T, the ANOVA-RNN is espoused on the input data $\alpha_{In} = \{\alpha_1, \alpha_2, \ldots\ldots \alpha_t\}$,which contains a hidden vector sequence $\hbar_{Lyr} = \{\hbar_1, \hbar_2, \ldots\ldots. \hbar_t\}$ along with the output vector sequence $\beta_{out} = \{\beta_1, \beta_2, \ldots\ldots.., \beta_t\}$. The HL is measured as,

$$\hbar_{Lyr} = \Im_{act}[wg_{\alpha\hbar}\alpha_t + wg_{\hbar\hbar}\hbar_{t-1} + Ba] \tag{20}$$

where, the weight matrices (e.g. the input-hidden weight value is specified as $wg_{\alpha\hbar}$ and the HL weight value is symbolized as $wg_{\hbar\hbar}$) is notated as wg, the bias vector is mentioned as Ba and the ANOVA radial basis kernel function is proffered as $\Im_{act}$. The expression for the ANOVA radial basis kernel function is formulated as,

$$\Im_{act} = \sum\nolimits_{\kappa=1}^{n} Exp(-\sigma(\alpha_X^{\kappa} - \alpha_Y^{\kappa})^2) \tag{21}$$

where, the dimensional inputs are notated as α_X and α_Y.

Step 2: Next, to estimate the VMs' efficiency, the output layer is accountable. By utilizing the sigmoid activation function (σ_s), the output layer is activated; then, it is estimated as,

$$\beta_{out} = \sigma_S[wg_{\alpha\beta}\hbar_t + Ba] \tag{22}$$

$$\alpha = wg_{\alpha\beta}\hbar_t + Ba \tag{23}$$

Step 3: The following equation is utilized to compute the sigmoid activation function.

$$\sigma_S(x) = \frac{1}{1 + \varepsilon^{-\alpha}} \tag{24}$$

Step 4: Subsequently, by calculating the difference between the original value α_a and the predicted value $\hat{\alpha}_p$, the loss value is analyzed, which is specified as,

$$Loss = (\alpha_a - \hat{\alpha}_p)^2 \tag{25}$$

Regarding the pre-historical functions, the VMs are trained effectively if the model's loss value is zero ($Loss = 0$). By updating the weight values, the backpropagation is performed if the loss value $Loss \neq 0$.

At last, the VM's resource utilization are determined effectively by the ANOVA-RNN in t seconds for every iteration, by this workflows are generated according to the resource utilization for the clustered task.

C. PHASE 3: SELECTION OF VIRTUAL MACHINES

After that, the most appropriate virtual machines to perform the clustered task is selected by the means of the LS-CSO algorithm. Here, the resources, as well as the capability of each virtual machine, are analyzed efficiently; thereby the suitable virtual machines are allocated for the execution of clustered tasks. Hence, this process drastically reduces resource consumption, and energy consumption and also speeds up the overall execution process.

1) SELECTION OF VIRTUAL MACHINES USING LS-CSO

CSO, which is centered on the intelligent behavior of crows, is a novel meta-heuristic optimization algorithm. The crow's idea of storing excess food in hiding places along with recovering it when required is a conception on which the CSO (a population-based approach) is centered [38]. In CSO, by deploying an LS algorithm, the awareness probability is enhanced in LS-CSO Algorithm.1. Thus, the optimal VM is chosen precisely. The following are the CSO's fundamental principles.

- Crows live in the form of a flock.
- Crows remember their hiding places.
- Crows flock together to steal.
- Crows keep their food stores from being stolen by a probability of thieves.

Step 1: Initialization

- Initially, the crows' population is initialized randomly in d-dimensional. N signifies the flock size. In this, every single crow denotes the VMs. The initialization process is expressed as,

$$Cw = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_d^1 \\ x_1^2 & x_2^2 & \cdots & x_d^2 \\ \vdots & \vdots & \vdots & \vdots \\ x_1^N & x_2^N & \cdots & x_d^N \end{bmatrix} \tag{26}$$

Each crow memory is initialized. It is supposed that the crows hide their foods at their initial positions since they have no experience at the initial iteration.

$$Mem = \begin{bmatrix} M_1^1 & M_2^1 & \cdots & M_d^1 \\ M_1^2 & M_2^2 & \cdots & M_d^2 \\ \vdots & \vdots & \vdots & \vdots \\ M_1^N & M_2^N & \cdots & M_d^N \end{bmatrix} \tag{27}$$

Step 2: Evaluate fitness (objective) function

To analyze every single crow, the fitness function is calculate; then, its value is assumed as an initial memory value. Every crow amasses its hiding place in its memory variablMem$_j$

Step 3: Generate new positio

The crow updates its position by selecting a random another crow, such thatx$_j$ and generating a random value. If this value is greater than the awareness probability AP, then crow x$_i$ will followx$_j$ to know the Mem$_j$. In order to improve the awareness probability the linear scaling technique is employed and that is given by.

$$g(h) = \frac{s(x) - \min s(x)}{\max s(x) - \min s(x)} \tag{28}$$

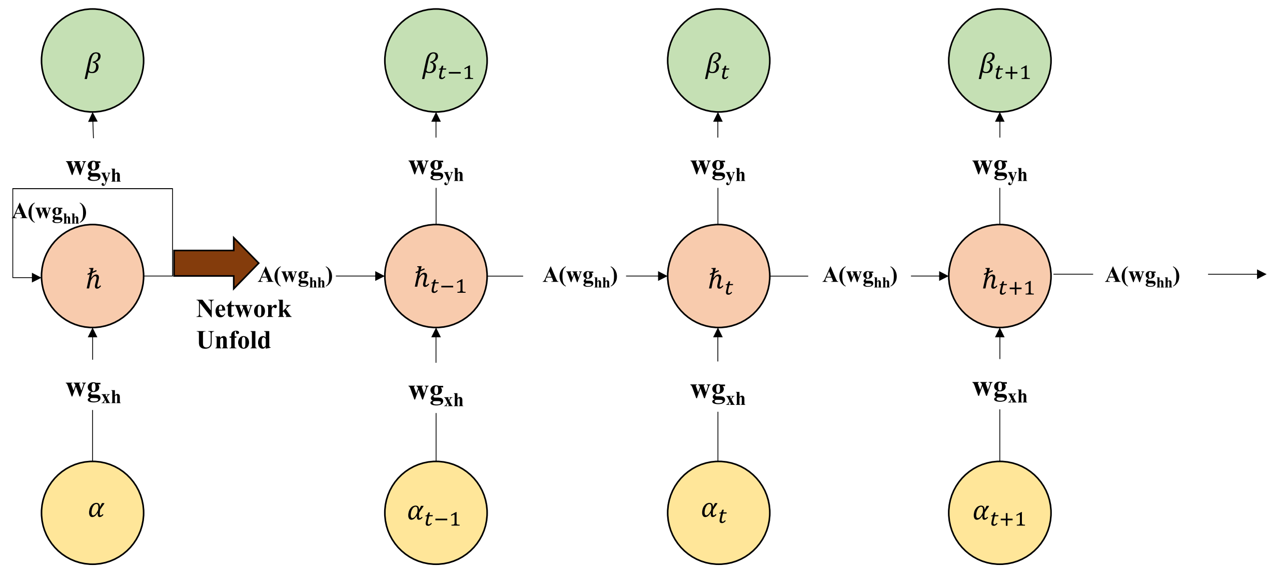

FIGURE 7. **Structure of A-RNN Classifier.**

Step 4: Updating to the new position

Crow updates its position by selecting a random another crow such thatx_j and following it to knowMem_j. Then, the new x_j is calculated as follows.

$$x_{i.Itr+1} = \left\{ \begin{array}{ll} x_{i,\mathrm{Itr}} + R_i \times Fl_{i,Itr} \times (Mem_{j,Itr} - x_{i,Itr}) & \mathrm{R}_j \geq AP_{j,Itr} \\ A\ random\ position & Otherwise \end{array} \right\} \tag{29}$$

where, $AP_{j,Itr}$ refers to crow j awareness probability,Itr refers to iteration number,R_i, R_j refers to random numbers,$Fl_{i,Itr}$ is the crow *i* flight length to denote crow j memory.

Step 5: Check the feasibility of new positions

Every single crow's probability of the new position is verified. The crow updates its position if its new position is feasible. Or else, the crow doesn't shift to the generated new position; also, it stays in the present position itself.

Step 6: Evaluate the fitness function of new positions for the new position of every single crow, the fitness function is calculated.

Step 7: Update memory

The memory is updated by the crow as,

$$Mem_{i,Itr+1} = \begin{cases} x_{i,Itr+1} & F(x_{i,Itr+1}) \leq F(Mem_{i,Itr}) \\ Mem_{i,Itr} & Otherwise \end{cases} \tag{30}$$

where, the objective function value is proffered as F. The crow updates its memory by the new position if the fitness function value of the crow's new position is better than the memorized position's fitness function value. Therefore, the optimal VMs are formulated as,

$$vm_{opt} = \{vm_1, vm_2, vm_3, \ldots\ldots.vm_n\} \tag{31}$$

2) JOB QUEUE

After the identification of the most appropriate virtual machines, the clustered tasks are arranged in a queue for execution. Thus, the tasks are scheduled to the virtual machines in such a way that the execution process consumes a minimal amount of energy.

IV. RESULTS AND DISCUSSIONS

Here, the complete evaluation of the proposed system's final outcome is illustrated. The performance along with the comparative evaluation is conducted to illustrate the efficiency of the work. By using Cloudsim, the proposed model is executed, and from the HCSP [39] and the GWA-T-12 Bitbrains datasets [40], which are publicly available on the internet.

A. PERFORMANCE ANALYSIS OF BLAKE

BLAKE2B performs well on 64-bit CPUs, on an Intel Core i7-11800H, BLAKE2B can process 1 gibibyte per second, the Figure 6. Demonstrates that BLAKE2B performed better than the SHA-1, BLAKE2S, MD5, SHA-512 on Intel CPUs.

Table 3. Determines the comparison of different services between the BLAKE2B and other Hash techniques. In case of BLAKE2B authentication is essential while uploading to data server and downloading the data from server, while other hashing techniques the authentication is not essential. It suggest that BLAKE2B is more faster which is assessed in Figure.8 and secure than other hash techniques

B. PERFORMANCE ANALYSIS OFMD-PAM

The proposed MH-PAM is evaluated in terms of Clustering time and outcomes are compared with various existing methods such as Mean shift, Fuzzy C Means, Kmeans, and PAM in order to state the worthiness of the method.

Algorithm 1 Pseudocode for LS-CSO Algorithm

```
Input: Number of virtual machines
Output: Selection of optimal virtual machines
Begin
    Initialize the positions of VM = {vm_1,vm_2,vm_3,.........vm_n}
    Evaluate the memory of each vm by the initial position
    Compute the fitness value for each vm
    Update the position of VM according to the random another vm
    Improve the awareness probability by using,
    $g(h) = \frac{s(x)-\min s(x)}{\max s(x)-\min s(x)}$
    If $R_j \geq AP^{j,Itr}$
        $x_{i,Itr} = x_{i,Itr} + R_i \times Fl_{i,Itr} \times (Mem_{j,Itr} - x_{i,Itr})$
    Else
        Relocate the vm randomly
    End if
    Check the feasibility of new position
    Update the VMs memory
    The process continue until the best solutions obtained
END
```

The Figure.9 illustrates the performance of MH-PAM, here four clustered groups are formed based on the length of the task to reduce the makespan efficiently.

In Figure 10, the proposed MH-PAM's cluster time is compared to that of other methods such as Means Shift, Fuzzy C Means, K-means, and PAM. Clustering time is simply the amount of time the classifier needs to cluster the tasks. By grouping one or more small tasks into a single job-style execution unit, clustering reduces make-span and energy consumption. This suggests that the proposed algorithm performs better than the current algorithm.

C. PERFORMANCE ANALYSIS OF PROPOSED A-RNN

Metrics such as precision, accuracy, specificity, sensitivity, recall, F1-Score, and training time are analysed for the proposed model. As shown in Table 4 and Table 5, these metrics are compared with the state of the art algorithms such as Deep Neural Network (DNN), RNN, Deep Belief Network (DBN), and Convolution Neural Network (CNN).

Regarding specificity, sensitivity, and accuracy, the performance evaluation of the A-RNN along with the existing approaches such as RNN, DBN, DNN, and CNN are shown in Table 4. The proposed A-RNN obtains high metric rates such as 97.62%, 97.60%, and 97.65%, for accuracy, sensitivity, and specificity; while the existing system achieves a low rate for accuracy, sensitivity, and specificity, which ranges between 93.66%-95.93%, 93.49%-95.81%, and 93.82%-96.05%, respectively. Thus, the proposed technique performs in a more secure manner along with manages the excess usage of energy.

A comparative evaluation of the metric values obtained by the proposed A-RNN along with the existing methodologies is illustrated in the Figure 11. If the system wants to be robust and effective, then the metrics values should remain high.

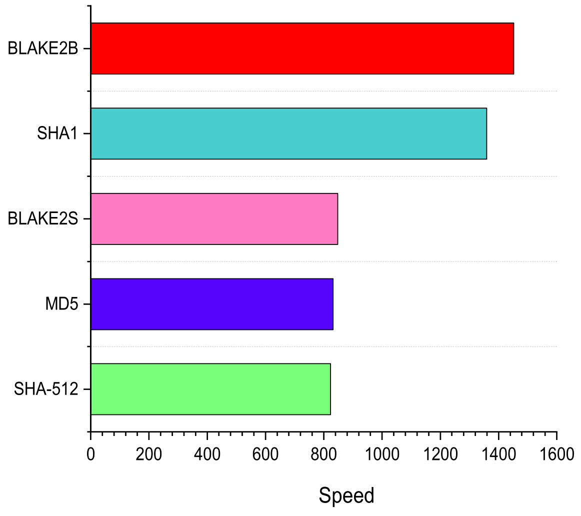

FIGURE 8. Various Hash functions speed.

Thus, when analogized to the existing RNN, DBN, DNN, and CNN, the proposed A-RNN technique guarantees higher accuracy, sensitivity, along with specificity rates. Thus, the proposed system gives accurate outcomes in WFS and manages the energy effectively.

In Table 5, the proposed A-RNN along with other existing RNN, DBN, DNN, and CNN models' performance metrics such as precision, recall, and FM are encompassed. By the high rate of precision, recall, and FM, the significance of the scheme is known. For precision, recall, along with FM, the proposed technique obtained 97.60%, 97.61%, and 97.59%; whereas the existing approach achieved the average of 92.43%, 92.67%, and 92.66%, which is comparatively low. Hence, the proposed A-RNN minimizes enormous complications and improves the steadiness of the WFS with enhanced security and low energy usage.

TABLE 3. Comparison table between BLAKE2B and other techniques.

Services	MD5	BLAKE -2B	SHA1	BLAKE - 2S	SHA-512
Privacy of data	Available	Available	Available	Available	Available
Integrity of data in the database	Available	Available	Available	Available	Available
Integrity of data in transit	Not available	Available	Not available	Available	Not available
While uploading data to a server, authentication is required	Not essential	Essential	Not essential	Not essential	Not essential
While downloading data to a server, authentication is required	Not essential	Essential	Not essential	Not essential	Not essential

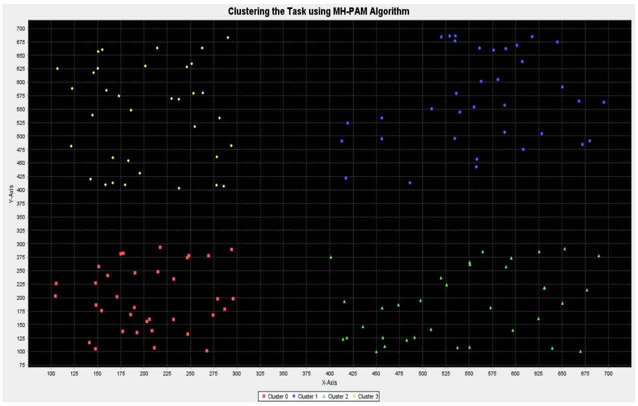

FIGURE 9. Task clustering performance by MD-PAM.

In Figure 12, the clear outlook of table 5 is shown. The comparative evaluation of the proposed methodology is exhibited in the figure 12. The proposed A-RNN achieves high performance ranges between 97.6% to 97.59% for precision, recall, and F-Measure, But for the existing metrics, the current RNN, DBN, DNN, and CNN achieve low performance between 89.96 % to 95.76%, Thus, the proposed A-RNN approach surpasses the other existing techniques and gives accurate outcomes under enormous complex conditions.

Training time taken by the proposed A-RNN is analogized with the prevailing RNN, DBN, DNN, and CNN in Figure 13. Training time is the quantity of time consumed by the classifier to train the data. The ET of the whole system will be affected if the system's training time is high. The system should consume less training time if it wants to be advanced. To complete the training process, the proposed A-RNN takes 54774 ms; while the prevailing techniques take 58987 ms for RNN, 61774 ms for DBN, 84774ms for DNN, and 94774 ms for CNN. Thus, compared to the existing approaches, the proposed A-RNN completes the whole training process with low computation time.

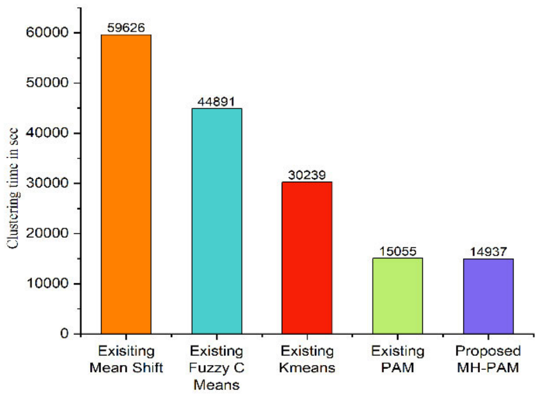

FIGURE 10. Comparative analysis of MH-PAM.

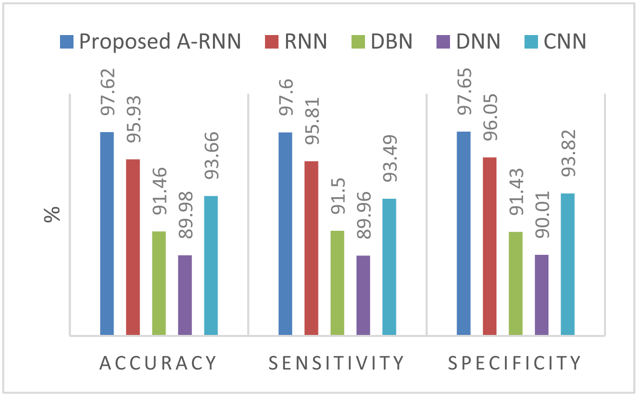

FIGURE 11. Graphical representation of the proposed A-RNN concerning the accuracy, sensitivity, and specificity.

TABLE 4. Performance evaluation of proposed A-RNN in terms of accuracy, sensitivity, and specificity.

Technique	Performance metrics (%)		
	Accuracy	sensitivity	specificity
Proposed A-RNN	97.62	97.60	97.65
RNN	95.93	95.81	96.05
DBN	91.46	91.50	91.43
DNN	89.98	89.96	90.01
CNN	93.66	93.49	93.82

Figure 14 compares the testing time of the proposed A-RNN to that of existing work such as RNN, DBN, DNN, and CNN. Testing time is nothing, but the time taken by the classifier to test the files. If the testing time is high, it degrades the performance of the model. The model is said to be best when testing is at its minimum. According to this, the proposed A-RNN takes 724 ms, whereas existing systems such as RNN, DBN, DNN, and CNN require 925 ms, 1233 ms, 1519 ms, and 1832 ms of testing time. Hence, the proposed A-RNN completes the entire testing process with less consumption time when compared to the existing work.

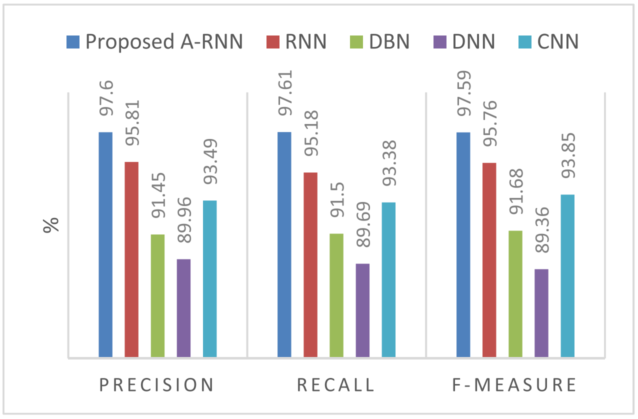

FIGURE 12. Graphical representation of the proposed A-RNN concerning the precision, recall, and F-Measure.

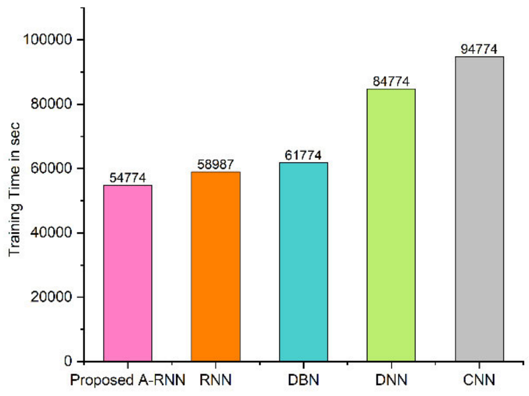

FIGURE 13. Comparison of the proposed A-RNN in terms of Training time.

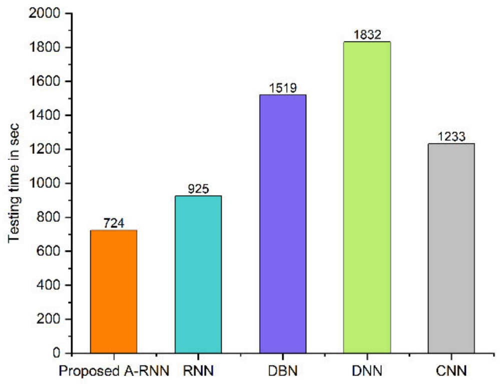

FIGURE 14. Comparison of the proposed A-RNN in terms of Testing time.

D. PERFORMANCE ANALYSIS OF PROPOSED LS-CSO

Regarding energy consumption, MS time, along with resource utilization, the LS-CS's performance is authenticated; then, the achieved outcomes ar

By utilizing the below equation, the total EC is estimate

$$E(x) = \sum En_{ij} + E_0 \tag{32}$$

$$subject\ to : \delta_j(Task_i) < T_{Deadline} \tag{33}$$

$$En_{ij} = er_j \times \delta_j(Task_i) \tag{34}$$

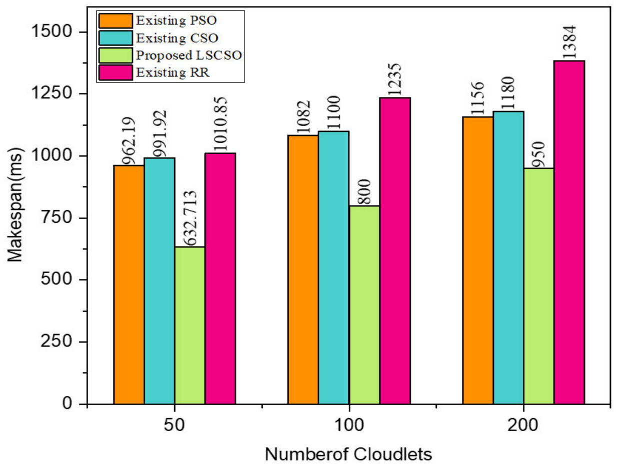

FIGURE 15. Make-span against 4 VMs.

TABLE 5. Performance evaluation of proposed A-RNN in terms of precision, recall, and F-measure.

Techniques	Performance metrics (%)		
	Precision	Recall	F-Measure
Proposed A-RNN	97.60	97.61	97.59
RNN	95.81	95.18	95.76
DBN	91.45	91.50	91.68
DNN	89.96	89.69	89.36
CNN	93.49	93.38	93.85

Here, the EC created by the tas T_i krunning on the VM, vm_j is denoted by $En_{i,j}$, the power required to operate a data center is represented by E_0, the EC rate of the VM is denoted by er_j, and the users' time constraint is represented by $T_{Deadlin}$.

The equation for determining MS is,

$$MinimizeMakespan = \min\{\max\{m_{v1}, \ldots\ldots.m_{vn}\}\} \tag{35}$$

$$m_{v1} = \sum_{i=0}^{n} \delta_j(Task_i) \tag{36}$$

$$\delta_j(Task_i) = \frac{l(Task_i)}{NPe_j \times Vmips_j}$$
$$\forall i = \{1, 2, 3, \ldots\ldots., n\},$$
$$j = \{1, 2, 3, \ldots\ldots, m\} \tag{37}$$

Here, the whole ET of a set of tasks running on vm_j is denoted by m_{v1}.The ET of the task i on vm_j is represented as $\partial_j(Task_i)$.The entire number of tasks is denoted by n and the number of vm is denoted by m, the length of the task in instruction MI is denoted by $l(Task_i)$.the number of processing elements is represented by NP_{ej}, and the vm speeds in million instructions per second ($mips$) is denoted by $Vmips_j$.

TABLE 6. Cloudsim experimental parameters.

Entity	Parameter	Values
VM	No. of VMs	4-8
	Band Width	1000 MBPS
	Size	10000 MB
	RAM	1024 MB
	VMM	Xen
	OS	Windows
	No. of CPU	1
	Policy	Time Shared
	No. of hosts	2
Host	Bandwidth	1000 MBPS
	RAM	2048 MB
	Storage	100000 GB
	Policy type	Time Shared
Data Centre	No. of data centres	2

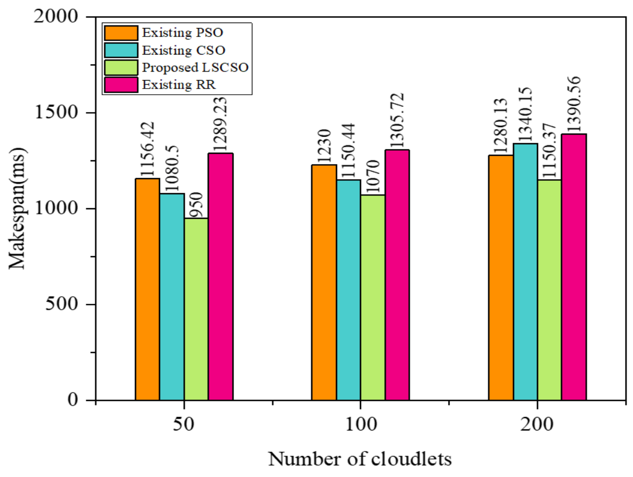

FIGURE 16. Make-span against 8 VMs.

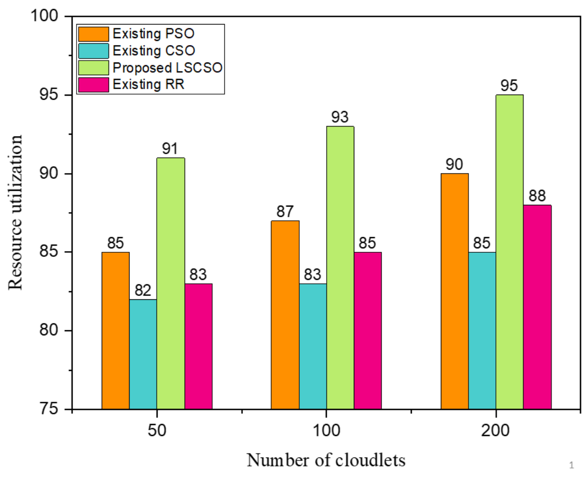

FIGURE 17. Resource utilization.

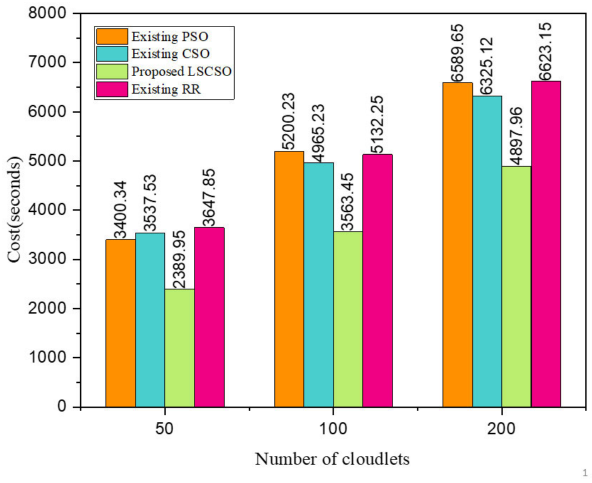

FIGURE 18. Cost against 4 VMs.

Finally, the quantity of resources used is calculated by the following equation.

$$CPU(i) = \frac{Total\ CPU\ usage\ of\ process\ (i)}{No\ of\ process(i)} \tag{38}$$

To evaluate the performance of proposed method cloudsim simulator [41] is used. Table.6 introduces the cloud platform components such as Datacenter, VMs and Hosts. These values are fixed for the simulation.

TheoutcomesofLS-CSOwerecomparedtothose of PSO,CSO and RR over a range of 50 to 200 cloudlets and 4 and 8VMs. Table6 displays the clouds imconfiguration with 2hosts,4 and 8VMs,2datacentres, and 50,100, and 200cloudlets,for which 40runs were applied to each scenario, and the average was then calculated.

1) MAKESPAN

Makespan is an important evaluation metric for determiningascheduler'seffectiveness. We have inputted 50, 100,200 number of cloudletson4&8VMsinsimulation environment to compare the proposed LS-CSO with existing algorithm. From Figures.15 and 16 observed that the LS-CSO outperformed PSO,CSO and RR.

2) RESOURCEUTILIZATION

From the perspective of the cloud provider, resource utilIsation is another critical metric in cloud computing. The resource utilization of proposed LS-CSO is compared with the existing PSO, CSO and RR algorithms in Figure.17 with 50,100, 200 cloudlets. It observed that 6% of better performance that the existing algorithms.

3) COST

The cost of task scheduling is related to various parameters, depending on the number of cloudlets and the number of available resources. It is more difficult to find the optimal global solutions when there are many tasks. As number of tasks increases the makespan, scheduling cost, waiting time are all increased due to insufficient existing resources. It is straightforward to arrive at a globally optimal solution when there are sufficient available resources. By comparison,

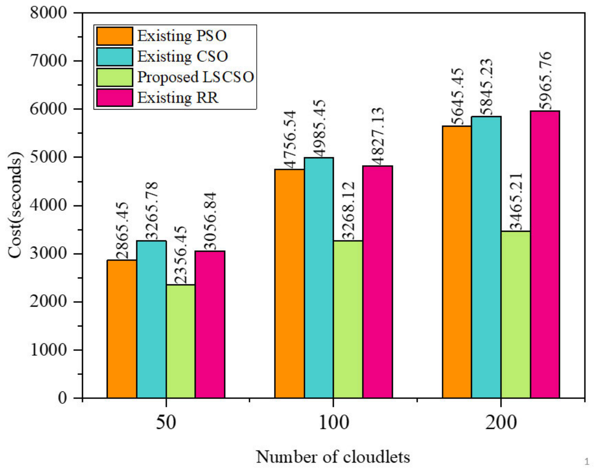

FIGURE 19. Cost against 8VMs.

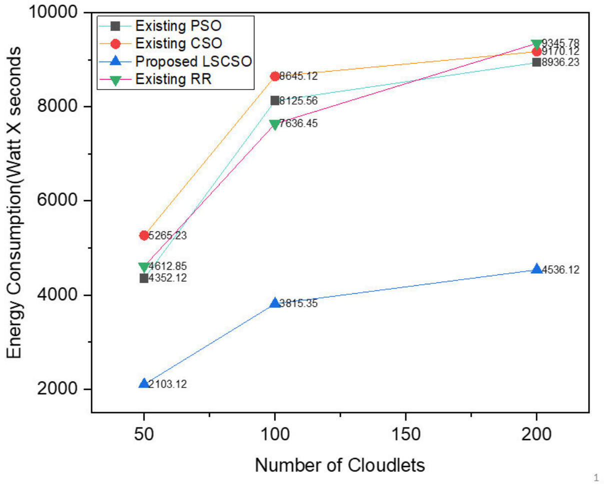

FIGURE 20. EnergyConsumption.

to CSO, PSO and RR the LS-CSO achieved 35%, 28%, and 29% respectively in the overall decreases execution cost as shown in Figure. 18&19.

4) ENERGY CONSUMPTION

For an LS-CSO algorithm with 8 virtual machines, the average energy consumption evaluation matrix shows that it consumes 52% less energy than CSO,PSO and RR algorithms as Shown in Figure.20. This is due to the overall execution time Of LS-CSO, which has as mall make-span, compared to CSO,PSO and RR algorithms that have higher make- spans. Because of this, these algorithms consumes more energy.

V. CONCLUSION

In this paper, a novel framework is proposed for scientific workflow scheduling in cloud computing. This frameworks targets authentication, energy consumption, resource utilization, cost, makespan as multi objective parameters. The framework composed in 3 phases, First Phase achieved authentication, which allows the user for secure scheduling using Blake2b with UUID, in second Phase the user tasks are initially clustered into different groups using MH-PAM, with the help of A-RNN resources monitored to generate dynamic workflow. In third Phase LS-CSO used for optimization to schedule the workflow on suitable VM's. To validate the efficiency of the proposed framework, comparative experimental results are presented. The outcomes demonstrated that the proposed framework performed better in terms of the multi-objective metrics taken into consideration than the current methods PSO and CSO. This analysis used the publicly available HCSP and GWA-T-12 Bitbrains datasets.

The work will be extended with some advanced neural networks and perform the workflow scheduling process in multiple clouds with real-time data.

REFERENCES

[1] 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, doi: 10.1016/j.jksuci.2020.12.001.

[2] Y. Gao, S. Zhang, and J. Zhou, "A hybrid algorithm for multi-objective scientific workflow scheduling in IaaS cloud," *IEEE Access*, vol. 7, pp. 125783–125795, 2019, doi: 10.1109/ACCESS.2019.2939294.

[3] Y. Xie, Y. Zhu, Y. Wang, Y. Cheng, R. Xu, A. S. Sani, D. Yuan, and Y. Yang, "A novel directional and non-local-convergent particle swarm optimization based workflow scheduling in cloud–edge environment," *Future Gener. Comput. Syst.*, vol. 97, pp. 361–378, Aug. 2019, doi: 10.1016/j.future.2019.03.005.

[4] Y. Zhao, Y. Li, I. Raicu, S. Lu, C. Lin, Y. Zhang, W. Tian, and R. Xue, "A service framework for scientific workflow management in the cloud," *IEEE Trans. Services Comput.*, vol. 8, no. 6, pp. 930–944, Nov. 2015, doi: 10.1109/TSC.2014.2341235.

[5] Z. Chen, K. Lin, B. Lin, X. Chen, X. Zheng, and C. Rong, "Adaptive resource allocation and consolidation for scientific workflow scheduling in multi-cloud environments," *IEEE Access*, vol. 8, pp. 190173–190183, 2020, doi: 10.1109/ACCESS.2020.3032545.

[6] W. Guo, B. Lin, G. Chen, Y. Chen, and F. Liang, "Cost-driven scheduling for deadline-based workflow across multiple clouds," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 4, pp. 1571–1585, Dec. 2018, doi: 10.1109/TNSM.2018.2872066.

[7] Jailalita, S. Singh, and M. Dutta, "Critical path based scheduling algorithm for workflow applications in cloud computing," in *Proc. Int. Conf. Adv. Comput., Commun., Autom. (ICACCA)*, Apr. 2016, pp. 1–6, doi: 10.1109/ICACCA.2016.7578905.

[8] Z. Zhu, G. Zhang, M. Li, and X. Liu, "Evolutionary multi-objective workflow scheduling in cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 5, pp. 1344–1357, May 2015, doi: 10.1109/TPDS.2015.2446459.

[9] F. Abazari, M. Analoui, H. Takabi, and S. Fu, "MOWS: Multi-objective workflow scheduling in cloud computing based on heuristic algorithm," *Simul. Model. Pract. Theory*, vol. 93, pp. 119–132, May 2019, doi: 10.1016/j.simpat.2018.10.004.

[10] Y. Wang, H. Liu, W. Zheng, Y. Xia, Y. Li, P. Chen, K. Guo, and H. Xie, "Multi-objective workflow scheduling with deep-*Q*-network-based multi-agent reinforcement learning," *IEEE Access*, vol. 7, pp. 39974–39982, 2019, doi: 10.1109/ACCESS.2019.2902846.

[11] J. Liu, J. Ren, W. Dai, D. Zhang, P. Zhou, Y. Zhang, G. Min, and N. Najjari, "Online multi-workflow scheduling under uncertain task execution time in IaaS clouds," *IEEE Trans. Cloud Comput.*, vol. 9, no. 3, pp. 1180–1194, Mar. 2019, doi: 10.1109/TCC.2019.2906300.

[12] Z. Li, V. Chang, H. Hu, H. Hu, C. Li, and J. Ge, "Real-time and dynamic fault-tolerant scheduling for scientific workflows in clouds," *Inf. Sci.*, vol. 568, pp. 13–39, Aug. 2021, doi: 10.1016/j.ins.2021.03.003.

[13] S. Z. M. Mojab, M. Ebrahimi, R. G. Reynolds, and S. Lu, "ICATS: Scheduling big data workflows in the cloud using cultural algorithms," in *Proc. 5th IEEE Int. Conf. Big Data Serv. Appl. (BigDataService)*, Apr. 2019, pp. 99–106, doi: 10.1109/BigDataService.2019.00020.

[14] H. Lu, X. Chen, Q. Liu, M. Samorani, G. Song, and Y. Yang, "Stochastic workflow authorizations with queueing constraints," *IEEE Trans. Depend. Secur. Comput.*, vol. 18, no. 4, pp. 1605–1619, Jul./Aug. 2021, doi: 10.1109/TDSC.2020.3026296.

[15] K. Kanagaraj and S. Swamynathan, "Structure aware resource estimation for effective scheduling and execution of data intensive workflows in cloud," *Future Gener. Comput. Syst.*, vol. 79, pp. 878–891, Feb. 2018, doi: 10.1016/j.future.2017.09.001.

[16] Y. Cui and Z. Xiaoqing, "Workflow tasks scheduling optimization based on genetic algorithm in clouds," in *Proc. IEEE 3rd Int. Conf. Cloud Comput. Big Data Anal. (ICCCBDA)*, Apr. 2018, pp. 6–10, doi: 10.1109/ICCBDA.2018.8386458.

[17] K. Dubey and S. C. Sharma, "An extended intelligent water drop approach for efficient VM allocation in secure cloud computing framework," *J. King Saud Univ., Comput. Inf. Sci.*, vol. 34, no. 7, pp. 3948–3958, Jul. 2022, doi: 10.1016/j.jksuci.2020.11.001.

[18] L. Zhang, L. Wang, Z. Wen, M. Xiao, and J. Man, "Minimizing energy consumption scheduling algorithm of workflows with cost budget constraint on heterogeneous cloud computing systems," *IEEE Access*, vol. 8, pp. 205099–205110, 2020, doi: 10.1109/ACCESS.2020.3037205.

[19] K. Dubey, M. Y. Shams, S. C. Sharma, A. Alarifi, M. Amoon, and A. A. Nasr, "A management system for servicing multi-organizations on community cloud model in secure cloud environment," *IEEE Access*, vol. 7, pp. 159535–159546, 2019, doi: 10.1109/ACCESS.2019.2950110.

[20] N. Hazekamp, N. Kremer-Herman, B. Tovar, H. Meng, O. Choudhury, S. Emrich, and D. Thain, "Combining static and dynamic storage management for data intensive scientific workflows," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 2, pp. 338–350, Feb. 2018, doi: 10.1109/TPDS.2017.2764897.

[21] A. Asghari, M. K. Sohrabi, and F. Yaghmaee, "A cloud resource management framework for multiple online scientific workflows using cooperative reinforcement learning agents," *Comput. Netw.*, vol. 179, Oct. 2020, Art. no. 107340, doi: 10.1016/j.comnet.2020.107340.

[22] H.-W. Kim, J. H. Park, and Y.-S. Jeong, "Human-intelligence workflow management for the big data of augmented reality on cloud infrastructure," *Neurocomputing*, vol. 279, pp. 19–26, Mar. 2018, doi: 10.1016/j.neucom.2017.04.082.

[23] H. K. Patnaik, M. R. Patra, and R. Kumar, "A workflow based approach for task scheduling in cloud environment," *Mater. Today, Proc.*, Aug. 2021, doi: 10.1016/j.matpr.2021.07.241.

[24] A. Belgacem and K. Beghdad-Bey, "Multi-objective workflow scheduling in cloud computing: Trade-off between makespan and cost," *Cluster Comput.*, vol. 25, no. 1, pp. 579–595, Feb. 2022, doi: 10.1007/s10586-021-03432-y.

[25] R. A. Haidri, C. P. Katti, and P. C. Saxena, "Cost effective deadline aware scheduling strategy for workflow applications on virtual machines in cloud computing," *J. King Saud Univ., Comput. Inf. Sci.*, vol. 32, no. 6, pp. 666–683, Jul. 2020, doi: 10.1016/j.jksuci.2017.10.009.

[26] E.-K. Byun, Y.-S. Kee, J.-S. Kim, E. Deelman, and S. Maeng, "BTS: Resource capacity estimate for time-targeted science workflows," *J. Parallel Distrib. Comput.*, vol. 71, no. 6, pp. 848–862, Jun. 2011, doi: 10.1016/j.jpdc.2011.01.008.

[27] I. Gupta, M. S. Kumar, and P. K. Jana, "Efficient workflow scheduling algorithm for cloud computing system: A dynamic priority-based approach," *Arabian J. Sci. Eng.*, vol. 43, no. 12, pp. 7945–7960, Dec. 2018, doi: 10.1007/s13369-018-3261-8.

[28] H. Aziza and S. Krichen, "A hybrid genetic algorithm for scientific workflow scheduling in cloud environment," *Neural Comput. Appl.*, vol. 32, no. 18, pp. 15263–15278, Sep. 2020.

[29] M. Karpagam, K. Geetha, and C. Rajan, "A reactive search optimization algorithm for scientific workflow scheduling using clustering techniques," *J. Ambient Intell. Hum. Comput.*, vol. 12, no. 2, pp. 3199–3207, Feb. 2021.

[30] H. R. Faragardi, M. R. S. Sedghpour, S. Fazliahmadi, T. Fahringer, and N. Rasouli, "GRP-HEFT: A budget-constrained resource provisioning scheme for workflow scheduling in IaaS clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 6, pp. 1239–1254, Jun. 2020.

[31] S. S. M. Nik, M. Naghibzadeh, and Y. Sedaghat, "Task replication to improve the reliability of running workflows on the cloud," *Cluster Comput.*, vol. 24, no. 1, pp. 343–359, Mar. 2021.

[32] K. K. Chakravarthi, L. Shyamala, and V. Vaidehi, "TOPSIS inspired cost-efficient concurrent workflow scheduling algorithm in cloud," *J. King Saud Univ., Comput. Inf. Sci.*, vol. 34, no. 6, pp. 2359–2369, Jun. 2022.

[33] J. K. V. Thekkepuryil, D. P. Suseelan, and P. M. Keerikkattil, "An effective meta-heuristic based multi-objective hybrid optimization method for workflow scheduling in cloud computing environment," *Cluster Comput.*, vol. 24, no. 3, pp. 2367–2384, Sep. 2021.

[34] H. Hafsi, H. Gharsellaoui, and S. Bouamama, "Genetically-modified multi-objective particle swarm optimization approach for high-performance computing workflow scheduling," *Appl. Soft Comput.*, vol. 122, Jun. 2022, Art. no. 108791.

[35] R. Garg, M. Mittal, and L. H. Son, "Reliability and energy efficient workflow scheduling in cloud environment," *Cluster Comput.*, vol. 22, no. 4, pp. 1283–1297, Dec. 2019.

[36] J. Zhou, J. Sun, M. Zhang, and Y. Ma, "Dependable scheduling for real-time workflows on cyber–physical cloud systems," *IEEE Trans. Ind. Informat.*, vol. 17, no. 11, pp. 7820–7829, Nov. 2021.

[37] S. Jananee, N. Annu, and K. Jayapriyadharshini, "Round rescheduling technique for hash function Blake using carry select adder with binary to excess converter," Coimbatore, India, Tech. Rep., 2013.

[38] A. G. Hussien, M. Amin, M. Wang, G. Liang, A. Alsanad, A. Gumaei, and H. Chen, "Crow search algorithm: Theory, recent advances, and applications," *IEEE Access*, vol. 8, pp. 173548–173565, 2020.

[39] *HCSP*. Accessed: Sep. 3, 2022. [Online]. Available: https://www.fing.edu.uy/inco/grupos/cecal/hpc/HCSP/

[40] *GWA-T-12 Bitbrains*. Accessed: Sep. 3, 2022. [Online]. Available: http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains

[41] R. 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. Exp.*, vol. 41, no. 1, pp. 23–50, Aug. 2011.

PILLAREDDY VAMSHEEDHAR REDDY received the B.Tech. degree in information technology and the M.Tech. degree in computer science and engineering from Jawaharlal Nehru Technological University, Telangana, India, in 2012 and 2014, respectively. He is currently pursuing the Ph.D. degree in computer science and engineering with VIT-AP University, Andhra Pradesh, India. He has six years of teaching experience. He has authored two research publications and one patent. His current research interests include cloud computing, computer networks, and machine learning. He is an active member of IAENG.

KARRI GANESH REDDY (Member, IEEE) received the B.Tech. degree in information technology from Andhra University, in 2007, the M.Tech. degree in computer science and engineering (information security) from NIT Rourkela, in 2010, and the Ph.D. degree from NITK, Surathkal. He has an overall experience of 12 years, out of which four years in research and eight years in the teaching field. He is currently an Associate Professor of Grade2-SCOPE with VIT-AP University, Andhra Pradesh. He is also the Coordinator of the Cyber Security Center of Excellence and the Program Chair of networking and security with VIT-AP-University. He has published more than 20 papers in international journals and conferences. He holds five patents. His research interests include algorithm analysis, cloud computing, wireless network security, cyber security, and forensics.