Intelligent Automation & Soft Computing
DOI: 10.32604/iasc.2023.031931
Article

Interpretive Structural Modeling Based Assessment and Optimization of Cloud with Internet of Things (CloudIoT) Issues Through Effective Scheduling

Anju Shukla[1], Mohammad Zubair Khan[2], Shishir Kumar[3,*], Abdulrahman Alahmadi[2], Reem Ibrahim A. Altamimi[2] and Ahmed H. Alahmadi[2]

[1]VIT Bhopal University, Bhopal, 466114, India
[2]Department of Computer Science and Information, Taibah University, Medina, Saudi Arabia
[3]Babasaheb Bhimrao Ambedkar University Lucknow, 226025, India
*Corresponding Author: Shishir Kumar. Email: dr.shishir@yahoo.com
Received: 30 April 2022; Accepted: 22 September 2022

Abstract: Integrated CloudIoT is an emerging field of study that integrates the Cloud and the Internet of Things (IoT) to make machines smarter and deal with real-world objects in a distributed manner. It collects data from various devices and analyses it to increase efficiency and productivity. Because Cloud and IoT are complementary technologies with distinct areas of application, integrating them is difficult. This paper identifies various CloudIoT issues and analyzes them to make a relational model. The Interpretive Structural Modeling (ISM) approach establishes the interrelationship among the problems identified. The issues are categorised based on driving and dependent power, and a hierarchical model is presented. The ISM analysis shows that scheduling is an important aspect and has both (driving and dependence) power to improve the performance of the CloudIoT model. Therefore, existing CloudIoT job scheduling algorithms are analysed, and a cloud-centric scheduling mechanism is proposed to execute IoT jobs on a suitable cloud. The cloud implementation using an open-source framework to simulate Cloud Computing (CloudSim), based on the job's workload, is presented. Simulation results of the proposed scheduling model indicate better performance in terms of Average Waiting Time (AWT) and makespan than existing cloud-based scheduling approaches.

Keywords: CloudIoT; cloud-computing; scheduling; IoT; workload

1 Introduction

Today, numerous devices, machines, and sensors are connected to the Internet to provide heterogeneous services. Any machine capable of transferring data over the Internet with a unique identity falls under the IoT [1]. By interconnecting the machines with the Internet, the machines become smarter. IoT devices can make decisions on their own. These devices are connected to some network devices and communicate with different network protocols. To make decisions independently, these devices record data at every instance. Therefore, several issues arise; the amount of data can be massive when it finally reaches the data center.

Secondly, latency can be a big issue when decisions are made in a real-time environment. Thirdly, combining many proliferation machines and devices together means exorbitant customization costs.

Cloud computing has an effective data management and computing framework to solve various IoT issues [2]. Cloud computing inherits the concept of concurrent engineering by executing tasks in parallel to improve various Quality of Service (QoS) metrics. The ubiquity and incremental growth of data over the Internet are needed for parallelism. According to a Windows Azure blog post, during the time span of June 2011–2012, they received 100,000 to 300,000 requests per second on a normal load and 150000–900,000 requests per second on crest loads [3]. There is a need for huge clouds to deal with peak loads. IoT-based multimedia jobs are outsourced for computation on a suitable cloud. Effective scheduling is important to handle the massive heterogeneous workload generated by IoT devices and benefit from the cost-based services provided by cloud computing. Real-world events can be handled more dynamically by integrating the Cloud with IoT (CloudIoT) [4].

However, integration of Cloud and IoT makes sense for various applications like healthcare, smart home, agriculture, smart city, smart mobility, and video surveillance. It helps to manage the remotely connected sensors for collecting data from other sensor nodes. But due to adverse technologies, there is a need to identify open issues related to Cloud and IoT. The significance and impact of various issues need to be analyzed for the CloudIoT framework. ISM establishes the contextual relationships among various issues in this paper.

1.1 Interpretive Structural Modeling (ISM)

ISM is a computer-aided mechanism for developing a graphical representation of complex systems. It is used to establish relationships among various variables that define an issue. ISM uses a structural model to identify, rank, and establish the inter-relationship among the variables. ISM's major significance is identifying the most influential factors in the CloudIoT paradigm.

1.2 Contributions

The major contributions of the paper are listed below:

- Various CloudIoT issues are identified through literature and analyzed using the ISM approach to identify the most influential issues in the CloudIoT scenario.
- A cloud-centric scheduling is proposed for heterogeneous IoT jobs to improve the performance of the CloudIoT framework in terms of a job waiting time and the length of time that elapses from the start of work to the end (makespan).

The rest of the paper is organised as follows. Section 2 briefly describes the reference architecture of the CloudIoT framework. Significant literature on CloudIoT is presented in Section 3. The detailed stepwise description of the proposed methodology and scheduling model is presented in Section 4. Section 5 compares the simulation and validation results of the proposed model to those of other state-of-the-art methods. Section 6 presents a conclusion followed by future directions to enhance the proposed scheduling model.

2 CloudIoT

CloudIoT is an innovative trend that connects and manages millions of devices in very cost-effective manners that are dispersed globally. Cloud can profit from IoT to deal with real-world things by sharing the pool of highly computational resources rather than having local servers or personal devices to handle applications [4]. Fig. 1 shows various media services offered by the cloud to access, store, and process data securely and cost-effectively.

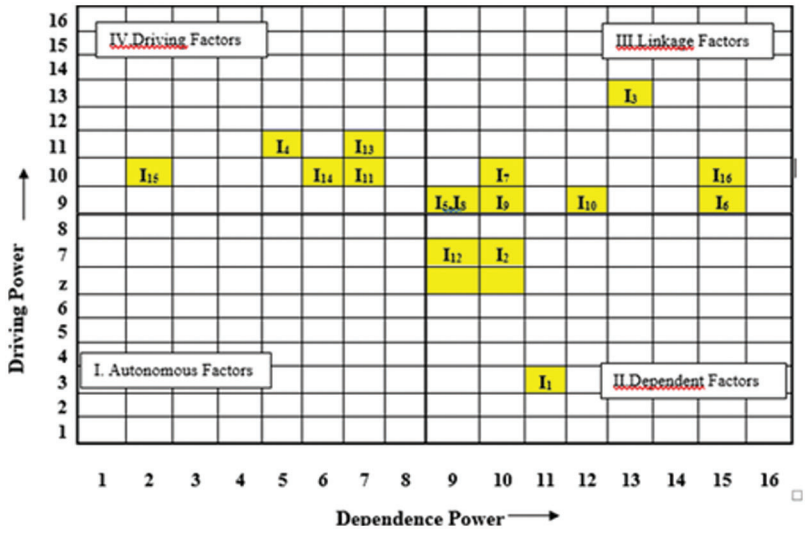

Figure 1: MICMAC analysis

CloudIoT Reference Architecture

Several components are involved in the CloudIoT reference architecture that is summarised as follows:-

(i) IoT devices-Any device or machine with a unique identity and capable of transferring data over the Internet comes under IoT.

(ii) Communication medium-A means is required to transfer data from these devices to the cloud to process the huge amount of data. A network with a protocol set is required to transfer data from these devices.

(iii) Device manager-Before sending data to the cloud, the device manager allows binding devices or establishing the relationship between IoT devices.

(iv) Access management-This service exposes the device over the Internet. Access management is responsible for authenticating objects that are accessing the network.

(v) Aggregation-In this step, the data is aggregated from all the devices for analysis and event processing.

(vi) Event processing and analytics-Some algorithms are set to clouds to handle events and take actions accordingly, such as temperature adjustment in smart homes. The data is collected, and analysis is done based on the condition.

(vii) Interface and Dashboard-The interface is used to visualise and analyse the collected data. The dashboard gives the layout of your device like energy consumption, heating or cooling system, and security alarm system at our homes, etc.

(viii) Application Program Interface (API)-The concept of API comes when data needs to be shared between two platforms. Platforms can change the data through APIs. For example, a message is sent to mobile whenever an event occurs.

3 Related Works

Some significant contributions in the field of the CloudIoT paradigm emphasize their particular research in this section. The literature focuses on issues and existing scheduling approaches in the CloudIoT paradigm.

3.1 Identification of CloudIoT Issues

Cloud computing enables access to various media services, resources, services, and infrastructure (e.g., Amazon, virtual private cloud, etc.) [5–7]. On the other hand, IoT can benefit from the cloud by using its unlimited storage and computation capabilities. Integrating both technologies generates new challenges and issues that need to be addressed.

Aazam et al. (2013) [8] presented an analysis of the expanding IoT and its integration with cloud computing. Various challenges and open issues of the CloudIoT framework are described. Data analysis, service provisioning, and storage are the future dimensions to improve the performance of the CloudIoT model.

Botta et al. (2016) [9] performed a survey and analysed the issues of integrating IoT and Cloud computing. IoT devices access data from the cloud for making decisions on their own, which is collected periodically through various sensors [10]. The periodic data collection mechanism leads to unnecessary energy consumption.

Dinh et al. (2017) [11] analyzed the sensing services and suggested a mechanism for data collection. The data is collected based on user demand and location. The cloud identifies the location and timing of user requirements – resulting in decreased energy consumption and increased network lifetime.

The limited literature on the CloudIoT paradigm is available and focused on its characteristics, technologies, challenges, and issues. Various CloudIoT issues have been identified and validated by industry professionals and are presented in Table 1. The identified issues are analyzed through the ISM approach and a proposed hierarchal model to identify the driving and dependency power among issues. However, no study analyzed the contextual relationships among the identified issues to the best of our knowledge, which is essential to formulate the relational model.

Table 1: Issues in CloudIoT

S. No	Parameter	Notation	Source
1	Device heterogeneity	I_1	Botta et al. (2015) [9]
2	Protocol type	I_2	Aazam et al. (2014) [8]
3	Cluster selection	I_3	Botta et al. (2015) [9]
4	Latency	I_4	Botta et al. (2015) [9]
5	Scheduling	I_5	Aazam et al. (2014) [8]
6	Data collection, security	I_6	Botta et al. (2015) [9]
7	Authenticity	I_7	Vijayasekaran et al. (2022) [10]
8	Resource selection	I_8	Aazam et al. (2014) [8]
9	Cost	I_9	Aazam et al. (2014) [8]
10	Job's workload	I_{10}	Vijayasekaran et al. (2022) [10]
11	Communication medium	I_{11}	Botta et al. (2015) [9]
12	Identity	I_{12}	Aazam et al. (2014) [8]
13	Access management	I_{13}	Botta et al. (2015) [9]
14	Confidentiality & data integrity	I_{14}	Botta et al. (2015) [9]
15	Storage	I_{15}	Aazam et al. (2014) [8]
16	Reliability	I_{16}	Botta et al. (2015) [9]

3.2 Scheduling in Cloud

As IoT devices produce a large amount of data, job scheduling is essential in CloudIoT-based applications [12,13]. Another reason for scheduling is outsourcing IoT jobs to various cloud resources. An efficient scheduling algorithm reduces the execution cost and improves the system throughput. Several task scheduling algorithms already exist in a cloud computing environment. Now, analyzing these approaches in the context of IoT applications is necessary. The reason behind this is that IoT devices and machines generate heterogeneous workloads. Resource availability is another issue due to the dynamic nature of cloud resources. Here, several kinds of literature on CloudIoT-based scheduling are briefly analyzed.

A heuristic-based task scheduling algorithm is presented for a cloud-fog computing environment by Pham et al. (2016) [14]. Mohanraj et al. (2021) [15] address the importance of scheduling policies in a cloud scenario. The effectiveness of single-cloud and multi-cloud-based scheduling is discussed to analyze the effectiveness. A multi-swarm optimization model for multi-cloud scheduling is presented to improve performance in a multi-cloud environment.

Lin et al. (2014) [16] considered memory usage, Central Processing Unit (CPU), and network bandwidth to implement task scheduling in cloud scenarios. The tasks are divisible and allocated to various resources for parallel execution. The algorithm gives a lower execution time than other state-of-the-art methods.

A hybrid optimization algorithm for resource selection is presented by Khan et al. (2021) [17]. The average waiting time is compared with existing approaches to show the model's effectiveness.

Sujana et al. (2019) [18] included security constraints in scheduling for scientific workflow applications. The method is capable of suitable resource selection for scientific workflow applications using the Particle Swarm Optimization (PSO) technique, which provides improved makesapn with other state-of-the-art methods.

To improve the performance of CloudIoT, Wu (2014) [19] designed a real-time task scheduling for embedded systems. These systems are mostly used for networks in which real-time constraints are associated. The algorithm improves the scheduling success rate of real-time tasks on heterogeneous processors.

The workload generated by IoT devices is analyzed and scheduled on the multi-cloud-based system by Moschakis et al. (2015) [20]. The global dispatcher selects the least loaded cloud for scheduling IoT jobs. Response time, service time, and job cost are analyzed to evaluate the algorithm's performance.

A novel Hybrid Task Scheduling Algorithm SJF and RR with dynamic quantum hybrid algorithm (SRDQ) considering a dynamic variable task quantum is presented by Elmougy et al. (2017) [21] to schedule jobs on available computing resources. The algorithm improves the SJF and RR cloud scheduling algorithms by considering the starvation conditions of long jobs due to the dynamic arrival of short jobs [22–24]. The algorithm performs better than SJF and RR in a cloud environment.

The paper's objective is to analyse CloudIoT issues through ISM to establish the interrelationship between them. This analysis helps us to identify the significant issues and propose effective scheduling for optimization of the CloudIoT paradigm in terms of makespan and average waiting time.

4 Methodologies

An analysis of the literature survey has been done, and a research gap has been identified. No framework exists for establishing the interrelationship among CloudIoT issues. This motivated us to identify the issues affecting the CloudIoT model's performance. ISM approach is used to establish the relationship among issues, which is appropriate to build up a hierarchical structure of various issues under investigation. The ISM classification results show that the schedule's driving and dependence power is higher than other

issues, so there is a probability that other issues will be highly influenced by scheduling (I5). Therefore, cloud-centric scheduling is proposed for IoT jobs to optimise the performance of the CloudIoT paradigm in terms of average waiting time and makespan. The following subsections describe the steps of the ISM approach and job scheduling model for CloudIoT.

4.1 Interpretive Structural Modeling (ISM) Approach

ISM is an interpretive analysis used to analyze the interrelationship among various issues associated with the framework. It provides a significant understanding of complex relationships by making a map among various elements involved in the scenario. The major aspects of the ISM approach are as follows: "I" symbolize the interpretive experience of experts. "S" indicates the structural bond among variables or issues. "M" defines the contextual relationship among issues through graphical representation. The Matriced' Multiplication Appliques a UN Classement (MICMAC) analysis is made to identify the driving and dependence power among issues.

The approach has been used worldwide in various applications, including software development, social science, supply chain management, decision-making, knowledge management, risk management, quality assurance, retail management, product management, etc.

ISM provides a structured model for the classification of various issues based on four factors-

- Autonomous factors are considered disengaged from the system or have no importance to the framework. Issues having less Driving Power (Dri_Pow) and less Dependence Power (Dep_Pow) come in this category.
- Dependent factors-These issues don't affect others but depend on other issues performance. The issues are having less Dri_Pow, but high Dep_Pow comes in this category.
- Linkage factors-The issues having high Dri_Pow and high Dep_Pow come in this category.
- Driving factors-The issues having high Dri_Pow but less Dep_Pow come in this category.

The detailed descriptions of each step are as follows:

4.1.1 Formation of Structural Self Interaction Matrix (SSIM)

To form the SSIM, close relationships among listed issues are analyzed based on expert opinion. A survey is formed to collect feedback from experts, and the output is based on the maximum response for the issue pair. Based on the relative relationship between issues Ii and Ij, various symbols (V, A, X, O) have been placed in each cell.

- V denotes that issue Ii influences issue Ij
- V denotes that issue Ij influences Parameter Ii
- X denotes that both issues Ii and Ij, both influences each other
- O denotes that both issues Ii and Ij, do not influence each other

SSIM is prepared and shown in Table 2 based on these rules.

4.1.2 Construction of Reachability Matrix

The SSIM is converted into an initial reachability matrix by placing 1 or 0 in each cell based on the following rules-

- If (Ii, Ij) contains V, then 1 is placed in (Ii, Ij) cell, and 0 is placed in (Ij, Ii) cell
- If (Ii, Ij) contains A, then 0 is placed in (Ii, Ij) cell, and 1 is placed in (Ij, Ii) cell
- If (Ii, Ij) contains X, then 1 is placed in both cells (Ii, Ij) and (Ij, Ii), respectively
- If (Ii, Ij) contains O, then 0 is placed in both cells (Ii, Ij) and (Ij, Ii), respectively

Table 2: Structural Self Interaction Matrix (SSIM)

I_i	I_{16}	I_{15}	I_{14}	I_{13}	I_{12}	I_{11}	I_{10}	I_9	I_8	I_7	I_6	I_5	I_4	I_3	I_2	I_1
I_1	O	A	O	O	O	O	A	O	O	O	O	V	O	O	O	
I_2	X	O	X	A	O	O	O	O	O	V	V	O	O	O		
I_3	V	O	O	O	V	O	A	X	X	A	O	X	O			
I_4	O	O	O	O	O	A	O	V	O	O	O	O				
I_5	O	O	O	O	O	O	X	X	X	O	X					
I_6	X	A	A	O	O	A	V	V	O	A						
I_7	X	O	A	X	X	X	O	O	O							
I_8	X	O	O	A	O	O	X	V								
I_9	O	A	O	O	O	O	X									
I_{10}	A	X	A	O	O	O										
I_{11}	X	O	O	O	O											
I_{12}	V	O	A	O												
I_{13}	V	A	V													
I_{14}	V	O														
I_{15}	V															
I_{16}																

The initial reachability matrix is converted into the final reachability matrix by removing the transitive dependencies among issues. The transitive dependency is found as follows-if issue Ii is related to Ij, and Ij is related to Ik, then issue Ii must be related to Ik. Table 3 shows the final reachability matrix for the selected issues.

Table 3: Final reachability matrix

I_i	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}	Dri_Pow
I_1	1	0	0	0	1	0	0	0	0	1*	0	0	0	0	0	0	3
I_2	0	1	0	0	0	1	1	0	0	0	0	1*	1*	1	0	1	7
I_3	1*	1*	1	0	1	1*	1*	1	1	1*	0	1	1	1*	0	1	13
I_4	1*	0	1*	1	0	1*	0	0	1	1*	1*	1*	1*	1*	0	1*	11
I_5	1*	0	1	0	1	1	0	1	1	1	1*	0	1	1	1	1*	12
I_6	0	1*	1*	0	1	1	1*	1*	1	1	0	0	0	0	0	1	9
I_7	0	1*	1	0	0	1	1	1*	0	0	1	1	1	1*	0	1	10
I_8	1*	1*	1	0	1	1*	0	1	1	1	0	0	0	0	0	1	9
I_9	0	0	1	1*	1	1*	0	1*	1	1	1*	0	0	0	0	1*	9
I_{10}	1	0	1	0	1	1*	0	1	1	1	0	0	0	0	1*	1*	9
I_{11}	1*	0	1*	1	1*	1	1	0	1*	0	1	1*	0	0	0	1	10
I_{12}	1*	1*	1*	0	0	1*	1	0	0	0	0	1	0	0	0	1	7

(Continued)

Table 3 (continued)

I_i	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}	Dri_Pow
I_{13}	1*	1	0	1*	1	1*	1	1	0	1*	1*	0	1	1	0	1	11
I_{14}	1*	1	1*	0	1	1	1	0	0	1	0	1	1*	1	0	1	10
I_{15}	1	1*	0	1*	1	1	1*	0	1	1	0	1*	1	0	1	1	10
I_{16}	0	1	1*	0	1*	1	1	1	1*	1	1	1*	0	0	0	1	10
Dep_Pow	11	10	12	5	12	15	10	9	10	12	7	9	7	6	2	15	

4.1.3 Level Partitioning of Final Reachability Matrix

Further reachability matrix is partitioned into different levels based on the intersection of Ri and Ai. The Ri contains the issues it may affect, and Ai contains the attributes that may influence it. Different iterations are performed to obtain the level partitioning as shown in Tables 4–10.

Table 4: First iteration

Issue	Ri	Ai	Ri∩Ai
I_1	I_1,I_5,I_{10}	I_1,I_3,I_4,I_5,I_8,I_{10},I_{11},I_{12},I_{13},I_{14},I_{15}	I_1,I_5,I_{10}
I_2	I_2,I_6,I_7,I_{12},I_{13},I_{14},I_{16}	I_2,I_3,I_6,I_7,I_8,I_{12},I_{13},I_{14},I_{15},I_{16}	I_2,I_6,I_7,I_{12},I_{13},I_{14},I_{16}
I_3	I_1,I_2,I_3,I_5,I_6,I_7,I_8,I_9,I_{10},I_{12},I_{13}, I_{14},I_{16}	I_3,I_4,I_5,I_6,I_7,I_8,I_9,I_{10},I_{11},I_{12},I_{13},I_{14}, I_{16}	I_3,I_5,I_6,I_7,I_8,I_9,I_{10},I_{12},I_{13},I_{14},I_{16}
I_4	I_1,I_3,I_4,I_6,I_9,I_{10},I_{11},I_{12},I_{13},I_{14}, I_{16}	I_4,I_9,I_{11},I_{13},I_{15}	I_4,I_9,I_{11}
I_5	I_1,I_2,I_3,I_5,I_6,I_8,I_9,I_{10},I_{11},I_{13}, I_{14},I_{15},I_{16}	I_1,I_2,I_3,I_5,I_6,I_8,I_9,I_{10},I_{11},I_{13},I_{14},I_{15}, I_{16}	I_1,I_2,I_3,I_5,I_6,I_8,I_9,I_{10},I_{11},I_{13},I_{14}, I_{15},I_{16}
I_6	I_2,I_3,I_5,I_6,I_7,I_8,I_9,I_{10},I_{16}	I_2,I_3,I_4,I_5,I_6,I_7,I_8,I_9,I_{10},I_{11},I_{12},I_{13}, I_{14},I_{15},I_{16}	I_2,I_3,I_5,I_6,I_7,I_8,I_9,I_{10},I_{16}
I_7	I_2,I_3,I_6,I_7,I_8,I_{11},I_{12},I_{13},I_{14},I_{16}	I_2,I_3,I_6,I_7,I_{11},I_{12},I_{13},I_{14},I_{15},I_{16}	I_2,I_3,I_6,I_7,I_{11},I_{12},I_{13},I_{14},I_{16}
I_8	I_1,I_2,I_3,I_5,I_6,I_8,I_9,I_{10},I_{16}	I_3,I_5,I_6,I_7,I_8,I_9,I_{10},I_{13},I_{16}	I_3,I_5,I_6,I_8,I_9,I_{10},I_{16}
I_9	I_3,I_4,I_5,I_6,I_8,I_9,I_{10},I_{11},I_{16}	I_3,I_4,I_5,I_6,I_8,I_9,I_{10},I_{11},I_{15},I_{16}	I_3,I_4,I_5,I_6,I_8,I_9,I_{10},I_{11},I_{16}
I_{10}	I_1,I_3,I_5,I_6,I_8,I_9,I_{10},I_{15},I_{16}	I_1,I_3,I_4,I_5,I_6,I_8,I_9,I_{10},I_{13},I_{14},I_{15},I_{16}	I_1,I_3,I_5,I_6,I_8,I_9,I_{10},I_{15},I_{16}
I_{11}	I_1,I_3,I_4,I_5,I_6,I_7,I_9,I_{11},I_{12},I_{16}	I_4,I_5,I_7,I_9,I_{11},I_{13},I_{16}	I_4,I_5,I_7,I_9,I_{11},I_{16}
I_{12}	I_1,I_2,I_3,I_6,I_7,I_{12},I_{16}	I_2,I_3,I_4,I_7,I_{11},I_{12},I_{14},I_{15},I_{16}	I_2,I_3,I_7,I_{12}
I_{13}	I_1,I_2,I_4,I_6,I_7,I_8,I_{10},I_{11},I_{13},I_{14}, I_{16}	I_2,I_3,I_4,I_7,I_{13},I_{14},I_{15}	I_2,I_4,I_7,I_{13},I_{14}
I_{14}	I_1,I_2,I_3,I_6,I_7,I_{10},I_{12},I_{13},I_{14},I_{16}	I_2,I_3,I_4,I_7,I_{13},I_{14}	I_2,I_7,I_{13},I_{14}
I_{15}	I_1,I_2,I_4,I_6,I_7,I_9,I_{10},I_{12},I_{13},I_{15}, I_{16}	I_{10},I_{15}	I_{10},I_{15}
I_{16}	I_2,I_3,I_5,I_6,I_7,I_8,I_9,I_{10},I_{11},I_{12},I_{16}	I_2,I_3,I_4,I_5,I_6,I_7,I_8,I_9,I_{10},I_{11},I_{12},I_{13}, I_{14},I_{15},I_{16}	I_2,I_3,I_5,I_6,I_7,I_8,I_9,I_{10},I_{16}

Table 5: Second iteration

Issue	Ri	Ai	Ri∩Ai
I_3	$I_3,I_7,I_8,I_{12},I_{13},I_{14}$	$I_3,I_4,I_7,I_8,I_{11},I_{12},I_{13},I_{14}$	$I_3,I_7,I_8,I_{12},I_{13},I_{14}$
I_4	$I_3,I_4,I_{11},I_{12},I_{13},I_{14}$	I_4,I_{11},I_{13},I_{15}	I_4,I_{11},I_{13}
I_7	$I_3,I_7,I_8,I_{11},I_{12},I_{13},I_{14}$	$I_3,I_7,I_{11},I_{12},I_{13},I_{14},I_{15}$	$I_3,I_7,I_{11},I_{12},I_{13},I_{14}$
I_8	I_3,I_8	I_3,I_7,I_8,I_{13}	I_3,I_8
I_{11}	$I_3,I_4,I_7,I_{11},I_{12}$	I_4,I_7,I_{11},I_{13}	I_4,I_7,I_{11}
I_{12}	I_3,I_7,I_{12}	$I_3,I_4,I_7,I_{11},I_{12},I_{14},I_{15}$	I_3,I_7,I_{12}
I_{13}	$I_4,I_7,I_8,I_{11},I_{13},I_{14}$	$I_3,I_4,I_7,I_{13},I_{14},I_{15}$	I_4,I_7,I_{13},I_{14}
I_{14}	$I_3,I_7,I_{12},I_{13},I_{14}$	$I_3,I_4,I_7,I_{13},I_{14}$	I_3,I_7,I_{13},I_{14}
I_{15}	$I_4,I_7,I_{12},I_{13},I_{15}$	I_{15}	I_{15}

Table 6: Third iteration

Issue	Ri	Ai	Ri∩Ai
I_3	I_3,I_7,I_{13},I_{14}	$I_3,I_4,I_7,I_{11},I_{13},I_{14}$	I_3,I_7,I_{13},I_{14}
I_4	$I_3,I_4,I_{11},I_{13},I_{14}$	I_4,I_{11},I_{13},I_{15}	I_4,I_{11},I_{13}
I_7	$I_3,I_7,I_{11},I_{13},I_{14}$	$I_3,I_7,I_{11},I_{13},I_{14},I_{15}$	$I_3,I_7,I_{11},I_{13},I_{14}$
I_{11}	I_3,I_4,I_7,I_{11}	I_4,I_7,I_{11},I_{13}	I_4,I_7,I_{11}
I_{13}	$I_4,I_7,I_{11},I_{13},I_{14}$	$I_3,I_4,I_7,I_{13},I_{14},I_{15}$	I_4,I_7,I_{13},I_{14}
I_{14}	I_3,I_7,I_{13},I_{14}	$I_3,I_4,I_7,I_{13},I_{14}$	I_3,I_7,I_{13},I_{14}
I_{15}	I_4,I_7,I_{13},I_{15}	I_{15}	I_{15}

Table 7: Fourth iteration

Issue	Ri	Ai	Ri∩Ai
I_4	I_4,I_{11},I_{13}	I_4,I_{11},I_{13},I_{15}	I_4,I_{11},I_{13}
I_{11}	I_4,I_{11}	I_4,I_{11},I_{13}	I_4,I_{11}
I_{13}	I_4,I_{11},I_{13}	I_4,I_{13},I_{15}	I_4,I_{13}
I_{15}	I_4,I_{13},I_{15}	I_{15}	I_{15}

Table 8: Fifth iteration

Issue	Ri	Ai	Ri∩Ai
I_{13}	I_{13}	I_{13},I_{15}	I_{13}
I_{15}	I_{13},I_{15}	I_{15}	I_{15}

Table 9: Sixth iteration

Issue	Ri	Ai	Ri∩Ai
I_{15}	I_{15}	I_{15}	I_{15}

Table 10: Final level partitioning

Issue	Ri	Ai	Ri∩Ai	Level
I_1	I_1,I_5,I_{10}	$I_1,I_3,I_4,I_5,I_8,I_{10},I_{11},I_{12},I_{13},I_{14},I_{15}$	I_1,I_5,I_{10}	I
I_2	$I_2,I_6,I_7,I_{12},I_{13},I_{14},I_{16}$	$I_2,I_3,I_6,I_7,I_8,I_{12},I_{13},I_{14},I_{15},I_{16}$	$I_2,I_6,I_7,I_{12},I_{13},I_{14},I_{16}$	I
I_3	I_3,I_7,I_{13},I_{14}	$I_3,I_4,I_7,I_{11},I_{13},I_{14}$	I_3,I_7,I_{13},I_{14}	III
I_4	I_4,I_{11},I_{13}	I_4,I_{11},I_{13},I_{15}	I_4,I_{11},I_{13}	IV
I_5	$I_1,I_3,I_5,I_6,I_8,I_9,I_{10},I_{11},I_{16}$	$I_1,I_3,I_5,I_6,I_8,I_9,I_{10},I_{11},I_{16}$	$I_1,I_3,I_5,I_6,I_8,I_9,I_{10},I_{11},I_{16}$	I
I_6	$I_2,I_3,I_5,I_6,I_7,I_8,I_9,I_{10},I_{16}$	$I_2,I_3,I_4,I_5,I_6,I_7,I_8,I_9,I_{10},I_{11},I_{12},I_{13},I_{14},I_{15},I_{16}$	$I_2,I_3,I_5,I_6,I_7,I_8,I_9,I_{10},I_{16}$	I
I_7	$I_3,I_7,I_{11},I_{13},I_{14}$	$I_3,I_7,I_{11},I_{13},I_{14},I_{15}$	$I_3,I_7,I_{11},I_{13},I_{14}$	III
I_8	I_3,I_8	I_3,I_7,I_8,I_{13}	I_3,I_8	II
I_9	$I_3,I_4,I_5,I_6,I_8,I_9,I_{10},I_{11},I_{16}$	$I_3,I_4,I_5,I_6,I_8,I_9,I_{10},I_{11},I_{15},I_{16}$	$I_3,I_4,I_5,I_6,I_8,I_9,I_{10},I_{11},I_{16}$	I
I_{10}	$I_1,I_3,I_5,I_6,I_8,I_9,I_{10},I_{15},I_{16}$	$I_1,I_3,I_4,I_5,I_6,I_8,I_9,I_{10},I_{13},I_{14},I_{15},I_{16}$	$I_1,I_3,I_5,I_6,I_8,I_9,I_{10},I_{15},I_{16}$	I
I_{11}	I_4,I_{11}	I_4,I_{11},I_{13}	I_4,I_{11}	IV
I_{12}	I_3,I_7,I_{12}	$I_3,I_4,I_7,I_{11},I_{12},I_{14},I_{15}$	I_3,I_7,I_{12}	II
I_{13}	I_{13}	I_{13},I_{15}	I_{13}	V
I_{14}	I_3,I_7,I_{13},I_{14}	$I_3,I_4,I_7,I_{13},I_{14}$	I_3,I_7,I_{13},I_{14}	III
I_{15}	I_{15}	I_{15}	I_{15}	VI
I_{16}	$I_2,I_3,I_5,I_6,I_7,I_8,I_9,I_{10},I_{11},I_{12},I_{16}$	$I_2,I_3,I_4,I_5,I_6,I_7,I_8,I_9,I_{10},I_{11},I_{12},I_{13},I_{14},I_{15},I_{16}$	$I_2,I_3,I_5,I_6,I_7,I_8,I_9,I_{10},I_{16}$	I

4.1.4 Cluster Matrix (MICMAC Analysis)

Fig. 1 shows the cluster matrix represents the interrelationship among various issues based on autonomous, dependent, linkage, and driving factors. The result shows that issue scheduling (I_5) has both the Dri_Pow and the Dep_Pow. Hence, effective scheduling of IoT jobs on a suitable cloud may improve the performance of the CloudIoT framework. Therefore, cloud-centric scheduling is proposed for the heterogeneous IoT jobs described in the next section.

4.2 Proposed Job Scheduling Model for CloudIoT Paradigm

The proposed scheduling model is designed to execute IoT jobs on various clouds based on their workload, as shown in Fig. 2. The job's workload is the amount of time a job needs a resource for computation. Each job is submitted to the job dispatcher through the gateway. A gateway is a doorway to the Internet. It communicates through Internet protocols and passes data to clouds for further processing.

The job dispatcher selects jobs in a first-come, first-serve manner. It interacts with the load collection module to know the current load of each cloud and selects the least loaded cloud to process the job. The load collection module sends the number of jobs in the job pool to inform the current load on the cloud. On each cloud, a job pool is maintained for arriving jobs.

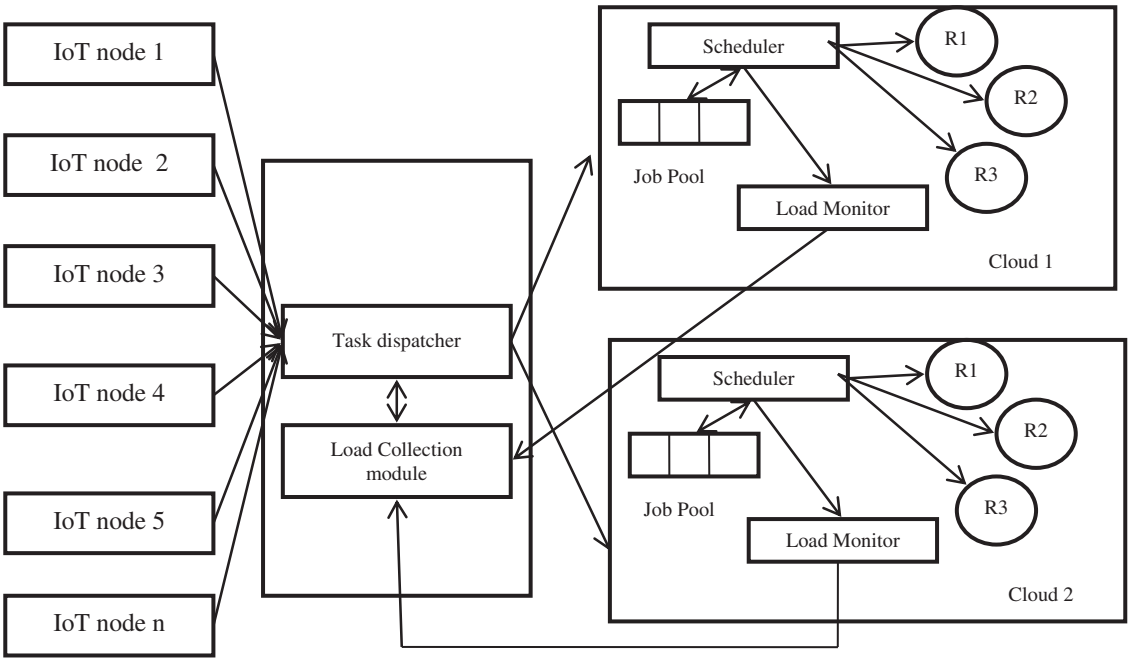

Figure 2: Job scheduling model for CloudIoT paradigm

The proposed algorithm is distributed in various modules and further integrated to frame the proposed scheduling model. The modules are as follows-

- Job dispatcher-Each IoT job is submitted to a job dispatcher to process on various clouds. It submits the job to the least loaded cloud for execution. Jobs are arranged in ascending order of arrival time.
- Load Collection Module-This module is responsible for collecting each cloud's current load and sending the information to the job dispatcher.
- Load monitor-on each cloud, the load monitor monitors the load whenever a job is submitted to a resource and periodically sends the information to the load collection module.
- Scheduler-At each cluster, a queue is maintained for jobs that are ready for execution. The scheduler selects a job and submits it to a resource in a First-Come, First-Served (FCFS) manner. The proposed algorithms I and II are used for job scheduling and quantum calculation on the selected resource.

4.2.1 Job Scheduling Algorithm

The objective of the job scheduling algorithm is to schedule heterogeneous jobs on suitable resources based on their workload. SJF ensures a minimum average waiting time, but long jobs may get starved as they never go for execution due to the dynamic arrival of short jobs. RR ensures a minimum average response time but produces an increased makespan. SRDQ considers the starvation condition but produces a high response time. To improve various QoS metrics, the algorithm splits jobs into two categories. If the job's workload is less than the average workload of ready queue jobs, it will be added

to Queue 1. Otherwise, the job is added to Queue 2. The idea behind this is to reduce response time and waiting time for jobs with a workload greater than average.

Algorithm I: Job Scheduling Algorithm

```
Input: Number of Jobs (J1, J2…Jn)
Output: Queue1, Queue 2
BEGIN
    1. for all jobs do
    2. Determine job with Arrival Time (AT) and Workload (WLD)
    3. end for
    4. for each job do
    5. while job exists do
    6. //Job categorization in Queue1 & Queue2
    7. While (! RQ_empty ()) do
    8.        for all jobs in RQ do
    9.              Sum = Sum + WLD[i];
    10.       end for
    11. average = sum/n;
    12. end while
    13. for all jobs in RQ do
    14. if (WLD[i] < average)
    15.       Queue 1 = J[i];
    16. else
    17.       Queue2 = J[i];
    18. end if
    19. end for
END
```

4.2.2 Quantum Calculation Algorithm

To process a job, all jobs are arranged in ascending order of workload. The dynamic quantum is calculated based on the median value of ready queue jobs. The median value is selected as a quantum value if jobs are odd. If the number of jobs is even, an average of two central numbers is taken as quantum. The following dynamic quantum calculation makes the algorithm more effective and less complex than SRDQ.

Algorithm II: Quantum Calculation Algorithm

```
Input: Number of jobs in job pool
Output: Quantum
BEGIN
    1. for all jobs do
    2.   for j = i + 1 to no of jobs
    3. if(J[j] < J[i])
    4.   //arrange in ascending order
    5.     Temp = J[i];
    6.     J[i] = J[j];
    7.     J[j] = temp;
    8.   end if
    9.   end for
    10. end for
    11. if (n % 2==0)
    12.   //number of jobs is even
    13. quantum = return (J [n/2] + J [n/2 – 1]/2);
    14. else
    15. //number of jobs is odd
    16. quantum = return J [n/2];
    17. end if
END
```

After quantum calculation, two jobs from queue-1 and one job from queue-2 are selected for execution. If the job finishes its execution, it leaves the queue and makespan, and Average Waiting Time (AWT) is calculated by Eqs. (1) and (2). If the job doesn't finish within the calculated quantum, Remaining Workload (RWLD) is computed by (3). Here, Start Time (ST), Arrival Time (AT), Completion Time (CT), and Workload (WLD) will be represented respectively. All remaining unfinished jobs are further processed in the shortest job first manner.

$$\text{makespan} = \frac{\sum_{j=1}^{n} CT_j - AT_j}{n} \tag{1}$$

$$AWT = \frac{\sum_{j=1}^{n} TAT_j - WLD_j}{n} \tag{2}$$

$$RWLD_j = WLD_j - Quantum_j \tag{3}$$

5 Simulation Analysis

The proposed scheduling model is simulated, and results are compared against RR, SRDQ, and Khan et al. (2021). To show the effectiveness of the proposed model, makespan and waiting time are evaluated

for all the algorithms. As discussed, IoT devices produce heterogeneous workloads, so simulation is performed on various data sets to measure the performance of the proposed scheduling model. The model is simulated on CloudSim (version 3.0.3) and consists of a single job dispatcher and three clouds. Each cloud consists of a single scheduler and three heterogeneous resources to process the incoming jobs. It is assumed that no more resources are added or left during the simulation. Workload range, arrival time range, number of jobs, IoT devices, and resources per cloud are given in Table 11. The various parameters that are used in the simulation are listed in Table 12. Various parameters (bandwidth, processing cost, memory cost, and storage cost) are the default values of the CloudSim toolkit. The main objective of the scheduling model is to select the least loaded cloud to process the job. So, at least three clouds are considered for effective cloud selection and fair results.

Table 11: Job categorization

Data set	No. of jobs	Workload range	Arrival time r ange (ms)	Number of IoT devices	Resources per cloud
Dataset 1	10	10–100	0–7	6	3
Dataset 2	100	100–200	0–7	10	3
Dataset 3	500	250–500	0–8	15	4
Dataset 4	1000	500–1000	0–8	20	4
Dataset 5	2000	1000–2000	0–20	25	5
Dataset 6	3000	2000–3000	0–30	30	5
Dataset 7	4000	3000–4000	0–40	35	6
Dataset 8	5000	4000–50000	0–50	40	6

Table 12: Simulation parameters

Parameters	Range
Number of IoT device	6–40
Number of clouds	3
Bandwidth (jobs/ms)	1000
Operating system	Windows 7
Processing cost ($)	2
Memory cost (MB)	0.5
Storage cost (MB)	0.001

Fig. 3 shows the AWT of RR, SRDQ, Khan et al. (2021), proposed algorithm *vs.* various data sets. The simulation result shows that the proposed algorithm decreased up to 39%, 34% and 30% of the AWT over RR, SRDQ, Khan et al. (2021) scheduling algorithms respectively. The reduction in waiting time shows that long jobs will also get service in time and will not be starved due to short and medium-sized jobs. The algorithm provides 500.4 ms AWT when the workload range is 250–500 rather than 820.4, 758.7, and 530.5 ms for the other algorithms.

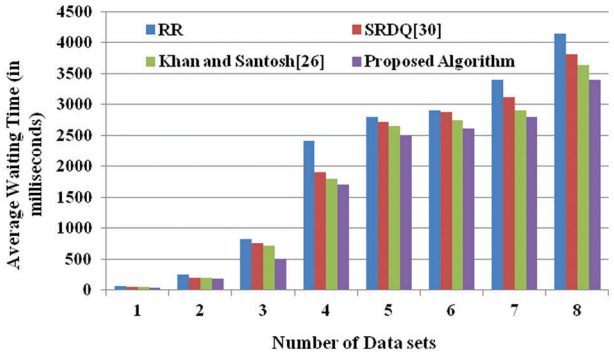

Figure 3: Average waiting time for various scheduling algorithms

Fig. 4 shows the makespan of RR, SRDQ [21], Khan et al. (2021) [17], and the proposed algorithm versus various data sets. The simulation result shows that the proposed algorithm provides the minimum makespan than RR, SRDQ [21], Khan et al. (2021) [17]. The proposed algorithm reduces upto 31.21%, 26.24%, and 11.97% of the makespan over RR, SRDQ and Khan et al. (2021) scheduling algorithms when the numbers of jobs are 2000. The algorithm provides 2741.7 ms makespan when the workload range is 1000–2000 rather than 3974.2, 3715.4 and 2979.8 ms for the RR, SRDQ and Khan et al. (2021) algorithms, respectively.

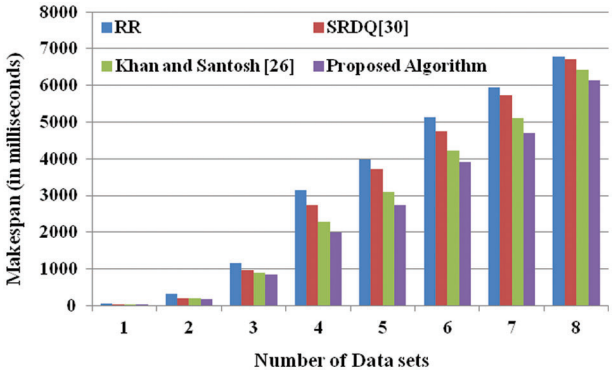

Figure 4: Makespan of various scheduling algorithms

6 Conclusions and Future Work

The resolution of CloudIoT generates new energizing directions for enterprises and researchers. Because IoT devices have limited processing and memory capacity, integrating the Cloud with IoT experiences is more valuable and limitless. This paper analyzes various CloudIoT issues through the ISM approach to establish the interrelationships among the identified issues. The ISM results show that issues Latency (I_4), communication medium (I_{11}), access management (I_{13}), confidentiality and data integrity (I_{14}), and storage (I_{15}) are strong driving issues and issues namely device heterogeneity (I_1), protocol type (I_2), and Identity (I_{12}) are dependent issues. The issues include cluster selection (I_3), scheduling (I_5), data collection and security (I_6), authenticity (I_7), resource selection (I_8), cost (I_9), job workload (I_{10}), and reliability (I_{16})have both Dri_Pow and Dep_Pow influence factors. Appropriate enhancement of these issues will result in the dynamic development of the CloudIoT framework.

The ISM analysis shows that scheduling is an important aspect and has both (driving and dependence) power to improve the performance of the CloudIoT model. Therefore, existing CloudIoT job scheduling

algorithms are analyzed, and a cloud-centric scheduling mechanism is proposed to execute IoT jobs on the suitable cloud. The proposed algorithm improves the performance of IoT applications by reducing waiting time and makespan. The algorithm performs better than RR, SRDQ and Khan et al. (2021) cloud scheduling algorithms. The proposed scheduling model can be further enhanced by computing energy consumption and adding budget and deadline constraints to the job to make the proposed model more effective.

Funding Statement: The author(s) received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] S. Goyal, S. Bhushan, Y. Kumar, A. U. H. S. Rana, M. R. Bhutta *et al.,* "An optimized framework for energy-resource allocation in a cloud environment based on the whale optimization algorithm," *Sensors*, vol. 21, no. 5, pp. 1583, 2021.

[2] S. Rani, D. Koundal, M. F. Ijaz, M. Elhoseny and M. I. Alghamdi, "An optimized framework for WSN routing in the context of industry 4.0," *Sensors*, vol. 21, no. 19, pp. 6474, 2021.

[3] J. Kaur, S. Ahmed, Y. Kumar, A. Alaboudi, N. Jhanjhi *et al.,* "Packet optimization of software defined network using lion optimization," *Computers, Materials & Continua*, vol. 69, no. 2, pp. 2617–2633, 2021.

[4] S. Singh and I. Chana, "A survey on resource scheduling in cloud computing: Issues and challenges," *Journal of Grid Computing*, vol. 14, no. 2, pp. 217–264, 2016.

[5] D. Kim, H. Kwon, C. Hahn and J. Hur, "Privacy-preserving public auditing for educational multimedia data in cloud computing," *Multimedia Tools and Applications*, vol. 75, no. 21, pp. 13077–13091, 2016.

[6] T. Ma, S. Pang, W. Zhang and S. Hao, "Virtual machine based on genetic algorithm used in time and power oriented cloud computing task scheduling," *Intelligent Automation and Soft Computing*, vol. 25, no. 3, pp. 605–613, 2019.

[7] Y. Li, J. Li, P. Shi and X. Qin, "Building an open cloud virtual dataspace model for materials scientific data," *Intelligent Automation and Soft Computing*, vol. 25, no. 3, pp. 615–624, 2019.

[8] M. Aazam, I. Khan, A. A. Alsaffar and E. N. Huh, "Cloud of things: Integrating internet of things and cloud computing and the issues involved," in *Proc. IBCAST*, Pakistan, pp. 414–419, 2014.

[9] A. Botta, W. D. Donato, V. Persico *et al.,* "Integration of cloud computing and internet of things: A survey," *Future Generation Computer Systems*, vol. 56, pp. 684–700, 2016.

[10] G. Vijayasekaran and M. Duraipandian, "An efficient clustering and deep learning based resource scheduling for edge computing to integrate cloud-IoT," *Wireless Personal Communications*, pp. 1–16, 2022.

[11] T. Dinh, Y. Kim and H. Lee, "A location-based interactive model of internet of things and cloud (IoT-cloud) for mobile cloud computing applications," *Sensors*, vol. 17, no. 3, pp. 489, 2017.

[12] A. Shukla, S. Kumar and H. Singh, "Fault tolerance based load balancing approach for web resources," *Journal of the Chinese Institute of Engineers*, vol. 42, no. 7, pp. 583–592, 2019.

[13] A. Shukla, S. Kumar and H. Singh, "Fault tolerance based load balancing approach for web resources in cloud environment," *International Arab Journal of Information Technology*, vol. 17, no. 2, pp. 225–232, 2020.

[14] X. Q. Pham and E. N. Huh, "Towards task scheduling in a cloud-fog computing system," in *2016 18th Asia-Pacific Network Operations and Management Symp.*, Japan, pp. 1–4, 2016.

[15] T. Mohanraj and R. Santhosh, "Multi-swarm optimization model for multi-cloud scheduling for enhanced quality of services," *Soft Computing*, pp. 1–11, 2021.

[16] W. Lin, C. Liang, J. Z. Wang and R. Buyya, "Bandwidth-aware divisible task scheduling for cloud computing," *Software: Practice and Experience*, vol. 44, no. 2, pp. 163–174, 2014.

[17] M. S. A. Khan and R. Santhosh, "Task scheduling in cloud computing using hybrid optimization algorithm," *Soft Computing*, pp. 1–11, 2021.

[18] J. A. J. Sujana, T. Revathi, T. S. Priya and K. Muneeswaran, "Smart PSO-based secured scheduling approaches for scientific workflows in cloud computing," *Soft Computing*, vol. 23, no. 5, pp. 1745–1765, 2019.

[19] D. H. Wu, "Task optimization scheduling algorithm in embedded system based on internet of things," *Applied Mechanics and Materials*, vol. 513, pp. 2398–2402, 2014.

[20] I. A. Moschakis and H. D. Karatza, "Towards scheduling for internet-of-things applications on clouds: A simulated annealing approach," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 8, pp. 1886–1899, 2015.

[21] S. Elmougy, S. Sarhan and M. Joundy, "A novel hybrid of shortest job first and round robin with dynamic variable quantum time task scheduling technique," *Journal of Cloud Computing*, vol. 6, no. 1, pp. 12, 2017.

[22] A. Shukla, S. Kumar and H. Singh, "An improved resource allocation model for grid computing environment," *International Journal of Intelligent Engineering and Systems*, vol. 12, no. 1, pp. 104–113, 2019.

[23] H. Singh and S. Kumar, "WSQ: Web server queueing algorithm for dynamic load balancing," *Wireless Personal Communications*, vol. 80, no. 1, pp. 229–245, 2015.

[24] N. Hasteer, A. Bansal and B. K. Murthy, "Assessment of cloud application development attributes through interpretive structural modeling," *International Journal of System Assurance Engineering and Management*, vol. 8, no. 2, pp. 1069–1078, 2017.