DEVELOPING AN APPROACH FOR SCHEDULING OF IOT APPLICATION TASKS IN FOG COMPUTING

Project report submitted in partial fulfilment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

By

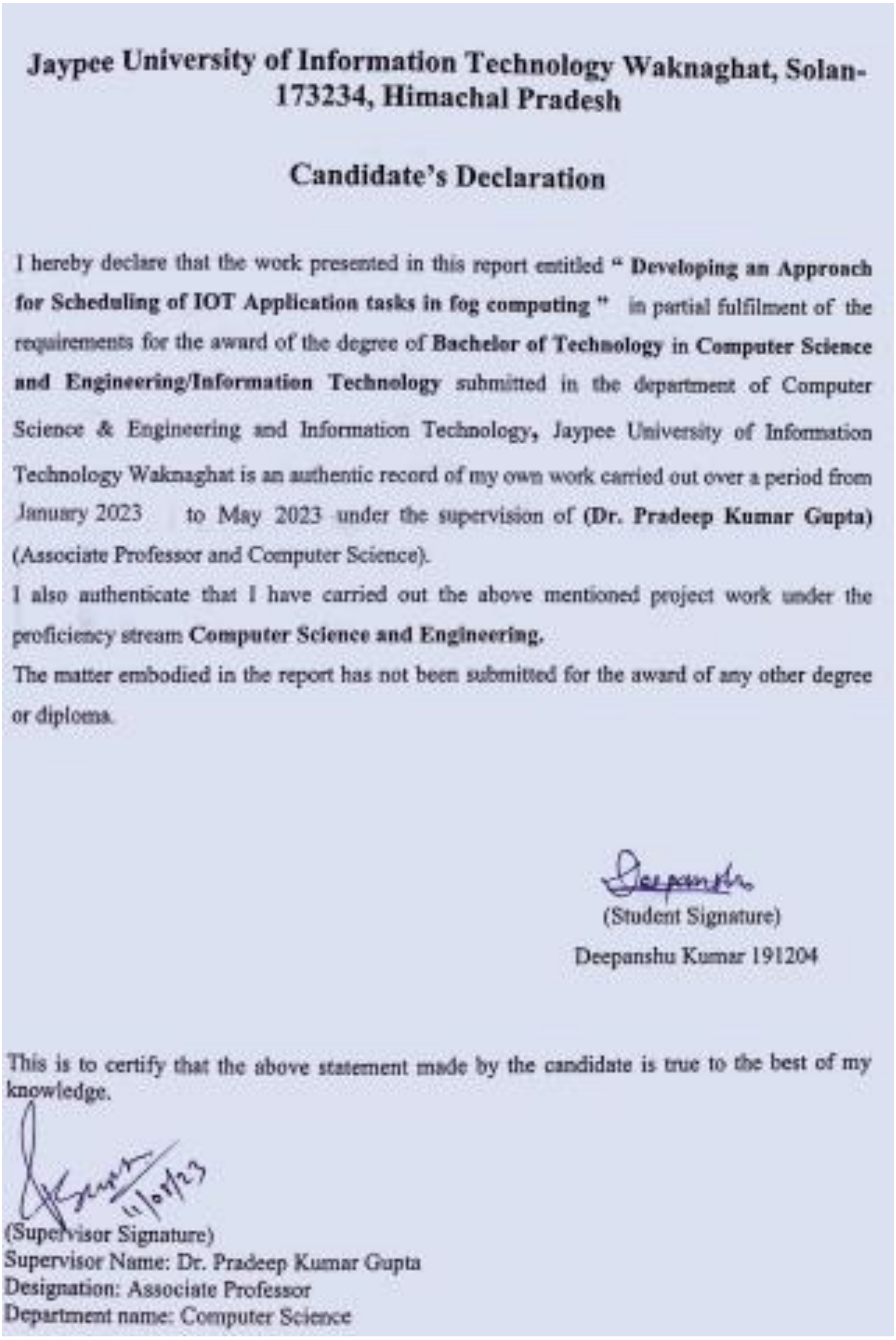
Deepanshu Kumar (191204)

Under the supervision of

Dr. Pradeep Kumar Gupta

to

Department of Computer Science & Engineering and Information Technology



Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

Candidate's Declaration

I hereby declare that the work presented in this report entitled " **Developing an Approach for Scheduling of IOT Application tasks in fog computing** " in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from January 2023 to May 2023 under the supervision of **(Dr. Pradeep Kumar Gupta)** (Associate Professor and Computer Science).

I also authenticate that I have carried out the above mentioned project work under the proficiency stream **Computer Science and Engineering.**

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

[illegible]

(Student Signature)

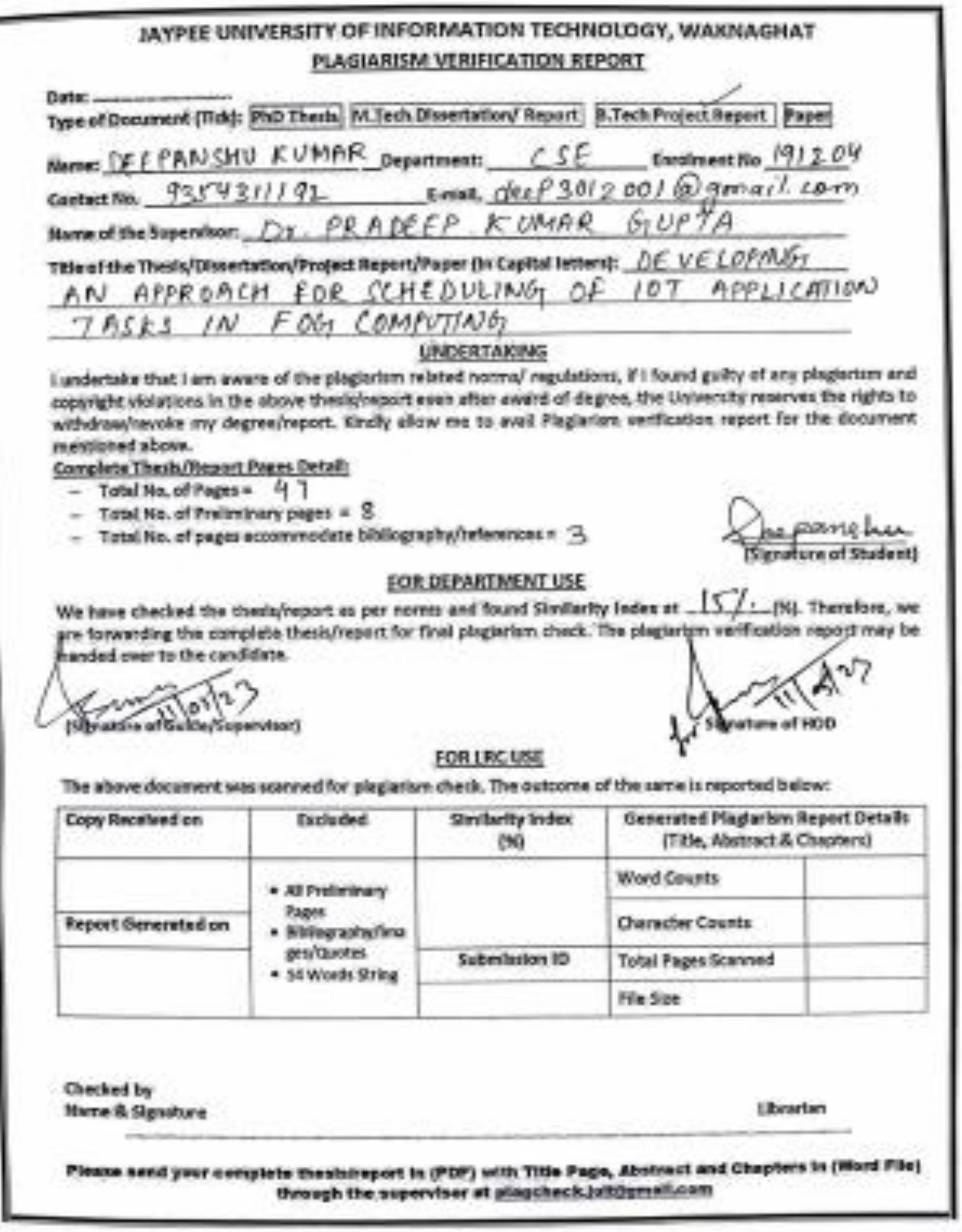
Deepanshu Kumar 191204

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

[illegible]

(Supervisor Signature)
Supervisor Name: Dr. Pradeep Kumar Gupta
Designation: Associate Professor
Department name: Computer Science

PLAGIARISM CERTIFICATE



JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: ....

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | [✓ B.Tech Project Report] | Paper

Name: DEEPANSHU KUMAR Department: CSE Enrolment No 191204

Contact No. 9354311192 E-mail. deep3012001@gmail.com

Name of the Supervisor: Dr. PRADEEP KUMAR GUPTA

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): DEVELOPING AN APPROACH FOR SCHEDULING OF IOT APPLICATION TASKS IN FOG COMPUTING

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages = 47
- Total No. of Preliminary pages = 8
- Total No. of pages accommodate bibliography/references = 3

[illegible]
(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found Similarity Index at 15% (%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

[illegible] 11/07/23
(Signature of Guide/Supervisor)

[illegible] 11/7/23
Signature of HOD

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	• All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature

Librarian

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

ACKNOWLEDGEMENT

I would like to thank and express our gratitude to our Project supervisor Dr. Pradeep Kumar Gupta for the opportunity that he provided us with this project "Developing an approach for Scheduling of IOT Application Tasks in Fog Computing". The outcome would not be possible without his guidance. This project taught me many new things and helped to strengthen concepts of Cloud Computing . Next, I would like to express my special thanks to the Lab Assistant for cordially contacting us and helping us in finishing this project within the specified time. Lastly, I would like to thank my friends and parents for their help and support.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

1. **PSO: particle swarm optimization**
2. **LA: learning automata**
3. **SJF: shortest job first**
4. **GA: genetic algorithm**
5. **HGAPSO: hybrid of genetic algorithm and particle swarm optimization.**

LIST OF FIGURES

LIST OF GRAPHS

ABSTRACT

Applications for the Internet of Things are now essential for raising living standards. However, the resources of conventional cloud data centres are under strain due to the growing volume of data produced by IoT devices.

The use of cloud computing is growing for large-scale Internet of things (IoT) applications that need a lot of computational power and storage. Fog computing brings cloud services to the network's edge. One of the significant challenges is to satisfy the quality of service requirements while assigning resources to tasks. In such a case, it is necessary to decide where applications should be executed in order to meet their quality of service requirements. As a result, a cloud system requires an efficient task scheduler to determine where applications should run. In this paper, we propose a hybrid approach for task scheduling by implementing a Hybrid Particle Swarm Optimization.

CHAPTER 1 : INTRODUCTION

Introduction

Numerous facets of our daily lives have been significantly impacted by the Internet of Things (IoT) and related technologies. It has made a variety of other items—not only conventional smart devices—able to connect to the internet. Vehicles, retail and logistics systems, traffic control systems, and health monitors are a few examples of these items that carry out a variety of tasks. It is vital to process the data generated by these items in order to acquire the information required for IoT applications.

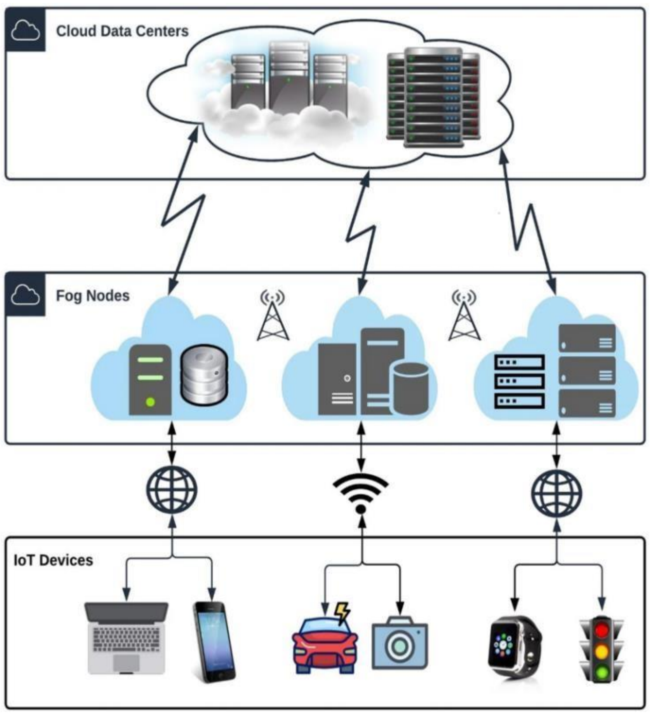


FIG 1. STRUCTURE OF CLOUD-FOG SYSTEM

IoT devices' ability to handle and store massive amounts of data is restricted. Additionally, network congestion and data transmission delays have a negative impact on the performance of time-sensitive IoT applications. Experts have expanded cloud computing resources to the edge of the network to address these problems.

By placing computer resources closer to end users, fog computing makes it easier to employ time-sensitive applications. In addition to many additional benefits (as shown in Figure 1), doing this helps preserve network capacity, lowers energy consumption, improves mobility and wide dispersion of IoT devices, and enables remote monitoring and low network latency for delay-sensitive IoT applications. [7]

Problem Statement

Task scheduling is one of several optimization-related problems that the IoT and cloud-fog networks face. This entails delegating an application's tasks to processing nodes at the network's edge, where it connects to the cloud. Depending on the resources at hand, a planner assigns tasks to an application in order to accomplish specified goals. The scheduler is in charge of deciding how resources should be used by programmes, taking into consideration variables relating to resource availability as well as application-specific elements like resource requirements and quality of service.

Enhancing the performance of fog computing requires effective resource management. Task scheduling is the process of allocating tasks to the appropriate resources in this context, and it is crucial in IoT systems for efficient resource management. It is crucial to implement realistic ways for allocating jobs in fog environments since the growing market for IoT devices places enormous processing demands on fog units.

Objectives

Large-scale IoT applications that need a lot of compute and storage capacity are increasingly turning to cloud computing as a solution. However, one of the major difficulties is making sure that the standards for service quality are met while effectively allocating resources to various tasks. Fog computing is a method for bringing cloud services to the edge of the network, however, it creates additional difficulties for resource allocation optimization. To ensure optimal system performance, a critical issue that must be solved is figuring out the best time to schedule a group of jobs on a fog node.

Finding the best moment to schedule a bunch of jobs on fog nodes is the goal of scheduling. This includes taking into account the service approach's two groups of scheduling parameters: service providers and consumer services. Fog node availability, processing power, storage capacity, and network bandwidth are just a few examples of the elements that the service provider must consider. Service level agreements (SLAs), quality of service (QoS) requirements, and task priority are examples of consumer service parameters. Fog computing systems may effectively distribute resources and guarantee that work requirements are met while maximising system performance by taking into account these scheduling criteria. Finding the best moment to schedule a bunch of jobs on fog nodes is the goal of scheduling. This includes taking into account the service approach's two groups of scheduling parameters: service providers and consumer services. Fog node availability, processing power, storage capacity, and network bandwidth are just a few examples of the elements that the service provider must consider. Service level agreements (SLAs), quality of service (QoS) requirements, and task priority are examples of consumer service parameters. Fog computing systems may effectively distribute resources and guarantee that work requirements are met while maximising system performance by taking into account these scheduling criteria.

Methodology

Fog Computing

A decentralized architecture known as fog computing is used to distribute data, compute, storage, and applications between the cloud and the data source. By locating computing resources close to the sites where information is generated and used, this technology, often referred to as edge computing, brings the advantages and capabilities of the cloud closer to the user. In Figure 2

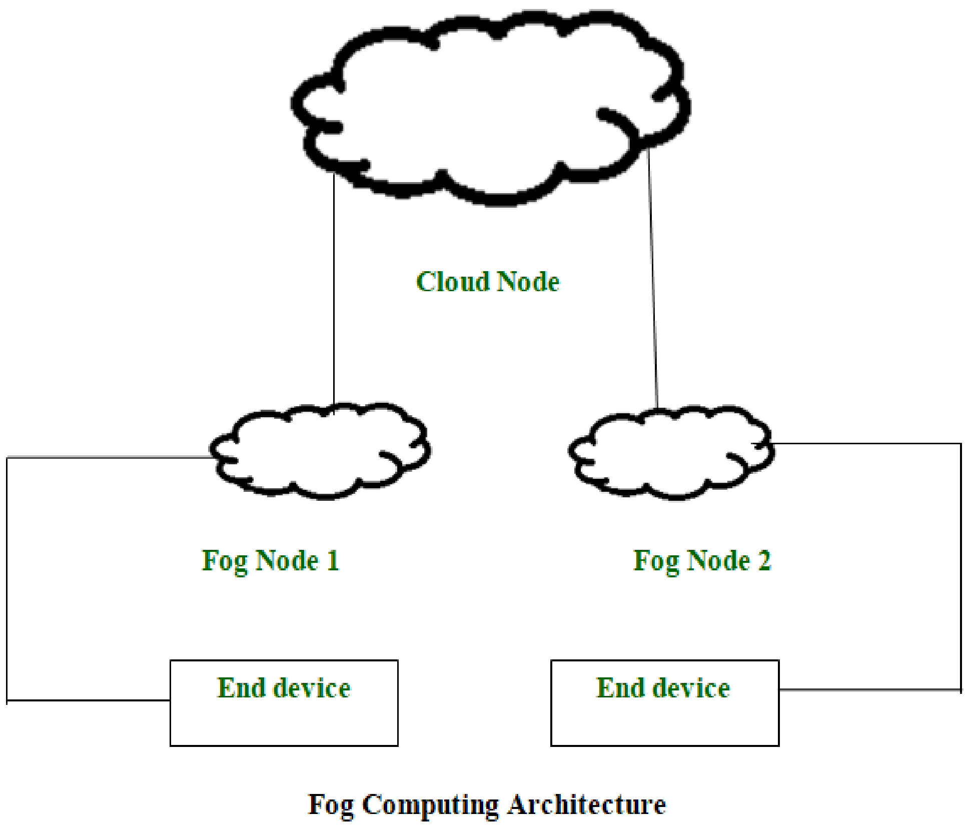


FIG 2. FOG COMPUTING ARCHITECTURE

The two types of task scheduling algorithms—distributed and centrally managed—can be applied to both homogeneous and heterogeneous resource systems. A single scheduler creates all mappings in centralised scheduling. Centralised scheduling has the benefit of being simple to set up, but if the scheduler fails, the entire system breaks down. Centralised scheduling also offers little failure tolerance. Distributed scheduling, in contrast, spreads jobs among several schedulers, allowing them to collaborate to match resources to task.

VIRTUAL MACHINE (VM)

The two types of task scheduling algorithms—distributed and centrally managed—can be applied to both homogeneous and heterogeneous resource systems. A single scheduler creates all mappings in centralised scheduling. Centralised scheduling has the benefit of being simple to set up, but if the scheduler fails, the entire system breaks down. Centralised scheduling also offers little failure tolerance. Instead, distributed scheduling divides up the workload across several schedulers, allowing them to collaborate on resource assignment.

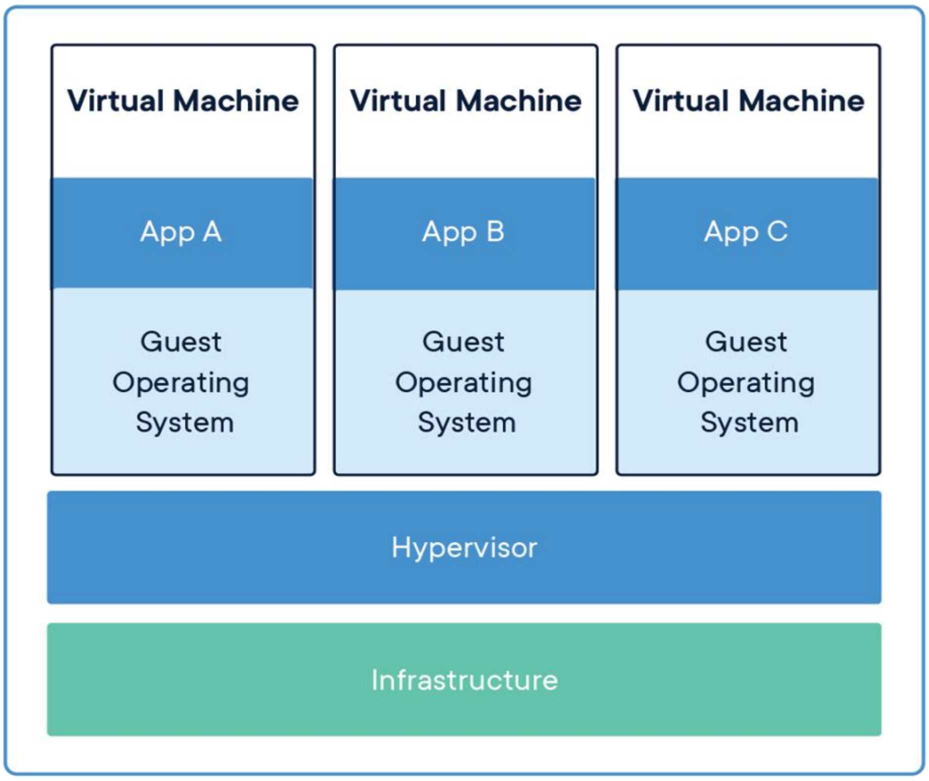


FIG 3. VIRTUAL MACHINE

A virtual machine (VM) contains crucial files like a log file, NVRAM setting file, virtual disc file, and configuration file. A VM operates as a process on an operating system. VMs are more substantial and take longer to boot than containers, but they provide benefits like distinct operating system kernels and conceptual separation between instances. They are perfect for running legacy software on outdated operating systems, decoupling apps, and running monolithic applications. Additionally, VMs and containers can be used together.

HPSOGA

Three mechanisms form the foundation of the HPSOGA technique. The discovery and utilisation processes are balanced by the first mechanism using Particle Swarm Optimisation (PSO). PSO is a population-based approach that was motivated by the crowding behaviour of birds, in which particles stand in for individuals and the population for a swarm.

Dimension reduction and population partitioning are a feature of the second HPSOGA mechanism. By dividing the solution into smaller groups, this approach hopes to make the solution space easier to search. This is accomplished by employing arithmetic cross operations in each group, which widens the algorithm's search space. This process contributes to the population's increased diversity, which may improve overall optimization.

The second mechanism in the HPSOGA method involves dimension reduction and population partitioning. This mechanism aims to break down the solution into smaller groups, allowing for a more efficient search of the solution space. This is achieved by using arithmetic cross operation in each group, which expands the search space for the algorithm. This step helps to increase the diversity of the population, which can lead to better global optimization..

SJF

Task scheduling in fog computing settings is frequently done using the scheduling algorithm SJF (Shortest Job First). The shortest task is scheduled first in this algorithm, which schedules tasks according to their execution times. By minimising the amount of time when resources are idle, SJF can also aid in optimising resource utilisation. SJF can assist keep resources in use and make sure they are not idle for long periods of time by prioritising the quickest jobs.

Because SJF prioritises tasks based on their execution time, which guarantees that the most crucial tasks are carried out first, it can be very effective in a fog environment. SJF is therefore perfect for fog contexts where activities frequently have different levels of urgency and priority.

If there are shorter jobs waiting in the queue, SJF may have the unintended consequence of delaying longer tasks. Preemptive SJF scheduling, which enables lengthier jobs to be interrupted if a shorter task with a higher priority enters the queue, can help to reduce this, though.

PSO

Popular metaheuristic optimisation algorithm Particle Swarm Optimisation (PSO) has been used for task scheduling in a variety of settings, including fog computing. By determining the ideal combination of fog nodes and resources to carry out each operation, PSO can be used to optimise task allocation and scheduling in fog environments.

PSO mimics the movement of a swarm of particles searching for the best answer as they move about in a search space. Each swarm particle in the context of task scheduling provides a potential remedy for the scheduling issue. Based on their own positions, the locations of the best solutions discovered thus far, and the locations of the best solutions discovered by the entire swarm, the particles move across the search space.

PSO has a number of benefits for task planning in foggy conditions. It may operate in dynamic contexts where the rate of job completion and the availability of resources can fluctuate. PSO is also simple to use and may be used to simultaneously optimise various goals, such reducing makepan and energy usage. Premature convergence is a difficulty that PSO may encounter where the algorithm becomes stuck in a poor solution. Hybrid PSO algorithms, which combine PSO with additional optimisation methods or heuristics, have been offered as a solution to this issue.

LEARNING AUTOMATA

A system called Learning Automata (LA) simulates the way a decision-making agent interacts with an unpredictably changing environment in order to improve over time. Artificial intelligence, control systems, and computer networks are just a few of the areas where LA has seen extensive use. LA offers a strong framework for decision-making processes that can be modified to varied scenarios and environments, making it a useful tool for resource scheduling.

The process of allocating resources to tasks in order to ensure their effective usage and prompt completion is known as resource scheduling. In many real-world applications, such as cloud computing, distributed systems, and Internet of Things (IoT) networks, it is a crucial problem. The scheduling problem is frequently difficult because it frequently calls for the optimisation of many goals, including minimising response time, maximising throughput, and reducing energy use.

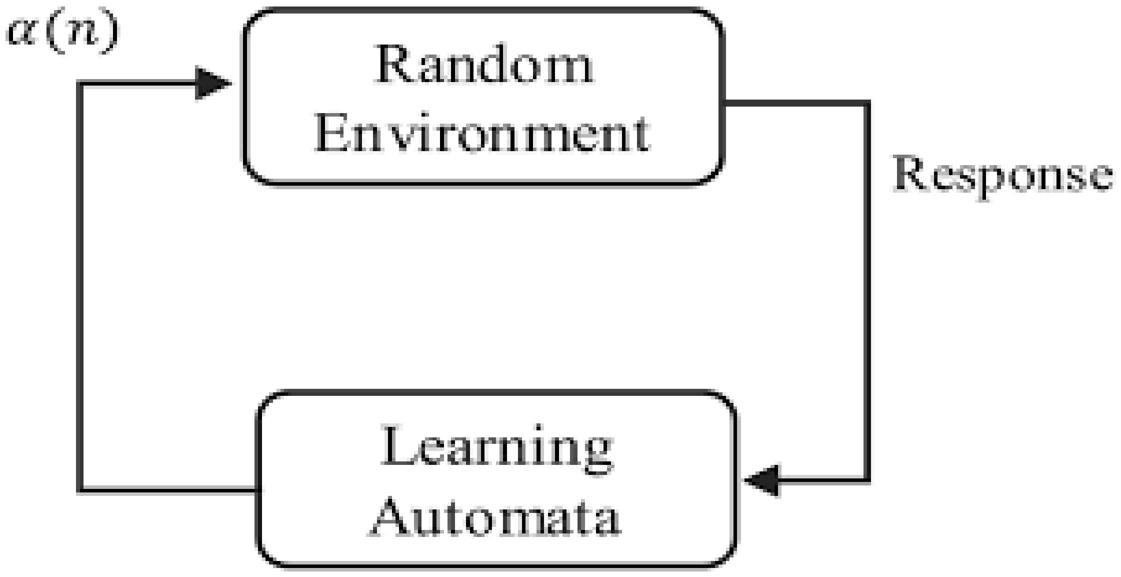


FIG 4 learning automata scheduling

By offering a framework for adaptive decision-making that can learn from experience and adjust to changes in the environment, LA can aid with resource scheduling. An agent in LA interacts with the environment and gets information based on what it does. The agent wants to discover a strategy that will improve its long-term success. By modelling the environment as a collection of states and actions, LA may be used to solve the scheduling problem. The agent then learns to choose the appropriate action based on the state of the environment.

One of LA's benefits is that it has the ability to learn in real-time, which is very helpful in circumstances that are unexpected and dynamic. LA is able to adjust to environmental changes including shifts in workload, variations in resource availability, or changes in user expectations. Additionally, LA can be used to optimise multiple goals because it can learn to balance various trade-offs based on feedback.

Organization

Chapter 1: Introduction

This section discusses a variety of project-related topics, including a project overview, the technique employed, a description of the issue the project aims to solve, and the project's goal.

Chapter 2: Literature Survey

This project's literature review section discusses the resources examined as well as the concepts discovered through study.

Chapter 3: System Development

We will discuss the project analysis and system design implementation in this section. We will go over the algorithms employed and give project snapshots.

Chapter 4: Performance Analysis

The performance analysis is presented in this area of the project by comparing and exhibiting the outcomes in the form of snapshots. The display of numerous outputs is another aspect of it. The part also discusses the methodology used to arrive at the outcomes and what significance they have for the project's success.

Chapter 5: Conclusion & Future work

The project's current work is concluded in this section, which also outlines potential directions for additional research and development.

CHAPTER 2 : LITERATURE REVIEW

It is important to note before we start discussing the literature on our subject that a wide range of academic scholars and paper authors have suggested study materials on picture reconstruction. These materials use a variety of approaches, filters, transforms, and spatial domain and transform-only combinations. We will give a quick overview of the pertinent research methodologies used by various scholars in this section.

A job scheduling strategy was suggested by Q. Liu et al.[1] for the fog-enabled Internet of Things (IoT) in smart cities. In order to schedule tasks in the fog environment, they designed a multi-objective optimisation method after analysing the characteristics and difficulties of IoT data processing. Through simulation experiments, they tested their suggested algorithm and compared it to two other ones already in use, demonstrating that their strategy could produce a better trade-off between task completion time and energy consumption. The authors also talked about how their strategy might be used in smart cities for things like traffic control and air quality monitoring.

A review of job scheduling strategies for fog computing was presented by Benchikh et al. [2]. these examined a number of variables that must be taken into account while job scheduling, including latency, energy consumption, and reliability, and how these affect the fog computing system's overall performance. They also contrasted the benefits and drawbacks of various task scheduling strategies, including heuristic algorithms, game theory, and artificial intelligence. The scientists stressed the significance of creating more effective and scalable task scheduling algorithms for fog computing as they offered potential future avenues for this field of study.

A real-time job scheduling method for Internet of Things (IoT) applications was put forth by P. Parimi et al.[3] in a fog-cloud computing environment. The suggested method employs a fuzzy logic-based mechanism to choose the best fog node for task offloading based on a number of factors, including task processing time, job priority, and resource availability. Through simulation experiments, the effectiveness of the suggested approach was demonstrated, and the results revealed that it performed better than current scheduling algorithms in terms of response time and energy consumption.

An overview of contemporary computing paradigms, such as cloud computing, IoT, edge computing, and fog computing, is given by K. De Donno et al. [4]. They analyse the development of these paradigms and emphasise their traits, benefits, and drawbacks. The study also discusses how these computing paradigms are being used in fields including healthcare, transportation, and smart cities. The paper's overall goal is to give readers a thorough knowledge of contemporary computer paradigms and their potential social effects.

A cost-aware job scheduling technique for fog-cloud situations is suggested by T.S. Nikoui et al. [5]. To ensure effective resource utilisation, the authors stress the significance of taking time and budget limits into account when scheduling tasks. They offer a heuristic approach that prioritises jobs that demand more resources and take longer to complete, while also accounting for the cost of running these jobs on cloud and fog nodes. Simulations are used to test the suggested algorithm, and the results show that it performs better in terms of cost savings and job completion times than the methods currently used for scheduling. In general, the study offers suggestions for improving task scheduling methods for fog-cloud situations.

For small-cell networks with multi-access edge computing (MEC), fuzzy-based collaborative task offloading was suggested by K.D. Hossain et al. [6]. To decide on the best offloading strategy, the method takes into account a number of factors, including job workload, battery state, channel quality, and the computing capacity of nearby tiny cells. The proposed method seeks to reduce energy usage and task offloading latency. To assess the scheme's performance in various settings, the authors ran comprehensive simulations. The simulation results demonstrated that the proposed strategy performs better in terms of energy usage and delay reduction than other cutting-edge alternatives. The proposed plan can help small-cell networks with MEC operate more effectively and efficiently overall.

A fuzzy logic-based emergency vehicle routing system for smart cities is proposed by R. R. Rout et al. [7]. The system uses real-time data gathered from multiple IoT sensors to pinpoint the emergency's position, the location of the closest hospital, and the current state of the roadways' traffic. The importance of the emergency is determined using a fuzzy inference technique, which is also utilized to design the best route for the emergency vehicle. Additionally, the system considers the current traffic conditions and modifies the route as necessary. The proposed system is put to the test in a mock smart city setting, and the findings demonstrate that it is successful in coming up with the best routes for emergency vehicles. The authors draw the conclusion that their method can be used in practical settings to speed up emergency service response times in smart cities.

A self-adapting work scheduling approach for container clouds was proposed by L. Zhu et al. [8] using learning automata. The technique is made to deal with the difficulty of effective resource utilisation and job scheduling in container cloud systems, where the demand for computational resources is highly dynamic.
The suggested algorithm decides on job scheduling and resource allocation using a learning automata-based method. The algorithm's learning automata component enables it to adapt to changing circumstances and gradually improve resource utilisation.They conducted simulated experiments utilising various workload scenarios to evaluate the suggested algorithm. In comparison to previous scheduling algorithms, the results demonstrate that the method is capable of achieving high resource utilisation and quick reaction times. The study shows that applying learning automata for work scheduling in container cloud systems is beneficial overall.

A learning automata-based Quality of Service (QoS) framework for Infrastructure-as-a-Service (IaaS) cloud environments was proposed by S. Misra et al. [8]. The framework's objective is to raise the quality of service for cloud services by making the best resource allocation choices..
The suggested framework employs a learning automata method to allocate resources in a dynamic manner in response to shifting service demand. Along with service level agreements (SLAs), resource availability, and user preferences and priorities, the framework is also made to consider these factors.

They evaluated the proposed framework through simulation experiments using different workload scenarios. The results show that the framework is able to effectively allocate resources and improve QoS compared to other resource allocation methods. The authors also analyze the scalability and overhead of the framework, and demonstrate its practical feasibility.

A learning automata-based scheduling approach for time-sensitive jobs in cloud environments was proposed by S. Sahoo et al. [10]. The suggested method aims to optimise task scheduling to ensure that all activities are completed by their due dates, while utilising the fewest resources possible and maintaining high standards of service. The suggested technique employs a learning automata method to dynamically modify the scheduling strategy in response to shifting job demands. The algorithm considers elements including task size, deadline, and priority, as well as the resources available and how they are being used. Through simulation experiments with a range of workload scenarios, the authors assess the proposed algorithm. The outcomes demonstrate that the suggested algorithm is capable of scheduling tasks efficiently, meeting their deadlines, and ensuring optimal resource utilisation and QoS.

Using learning automata, A. Valkanis et al.[11] suggest a reinforcement learning-based strategy for traffic prediction in core optical networks. The suggested method aims to increase traffic prediction accuracy, which is crucial for optimising resource allocation and raising QoS in optical networks. A. Valkanis et al.[11] propose a reinforcement learning-based method for traffic prediction in core optical networks using learning automata. The proposed approach seeks to improve the accuracy of traffic prediction, which is essential for improving resource allocation and enhancing QoS in optical networks.

A learning automata-based technique for load scheduling in power systems was put forth by Syed Q. Ali et al. [12]. The suggested method aims to meet the power demand while maximising resource allocation for power generation, lowering operational costs, and ensuring system stability.

The suggested method employs a learning automata algorithm to dynamically modify the generation plan in response to the fluctuating power demand and the accessibility of generation resources. The algorithm considers a number of variables, including the cost of generation, ramp rate, and minimal up/down times, as well as the demand for electricity and the accessibility of renewable energy sources.

Through simulation experiments using a realistic power system model, they assessed the suggested strategy. In comparison to alternative scheduling techniques, the results demonstrate that the suggested strategy may successfully optimise the generation schedule and lower operational costs. The impact of several aspects, such as the degree of renewable energy source penetration and the cyclicality of the power demand, on the performance of the suggested approach is also examined by the authors.

Sr.no.	Author(s)	Advantages	Disadvantages
1.	Q.Liu, Y. Wei, S. Leng and Y. Chen .[1]	Simulations are used to assess the method, enabling testing in a controlled setting.	When scheduling tasks, the algorithm does not take security and privacy concerns into account.
2.	K. Benchikh and L. Louail.[2]	Improved latency: Tasks may be processed closer to the edge with fog computing, which lowers latency and speeds up response times.	Security risks: The likelihood of security breaches or unauthorised access increases as data is processed and stored across several nodes.
3.	H.S. Ali, R. R. Rout, P. Parimi and S. K. Das [3]	The proposed approach can be easily integrated with existing fog-cloud computing frameworks.	The experimental evaluation is restricted to a single use case and might not be transferable to others.

Sr.no.	Author(s)	Advantages	Disadvantages
4.	K. De Donno, K. Tange and N. Dragoni[4]	raised awareness of the value of fog computing as a link between cloud and edge computing.	The challenges and constraints of each computer paradigm are not thoroughly examined in the paper.
5.	T.S. Nikoui, A. Balador, A. M. Rahmani and Z. Bakhshi[5]	Evaluation of the procedure in a simulated environment can be used to gauge its effectiveness and practicality.	The method may need extensive computation and data processing, which could increase the overall time required to complete jobs.
6.	K.D. Hossain, T. Sultana, V. Nguyen, T. D. Nguyen, L. N. Huynh, E.-N. Huh [6]	In terms of energy usage and delay, the plan performs better than other existing plans.	The suggested plan can be challenging to put into practise and complex.

Sr.no.	Author(s)	Advantages	Disadvantages
7.	R. R. Rout, S. Vemireddy, S. K. Raul and D. Somayajulu [7]	The suggested system has the ability to swiftly determine the best paths for emergency vehicles in real-time, which can help save crucial time in an emergency.	It is unknown how well the suggested method operates in use because the publication does not include a thorough examination of it.
8.	L. Zhu, K. Huang, Y. Hu and X. Tai[8]	The algorithm can gain knowledge from prior experiences and develop better decision-making over time by using learning automata.	The usefulness of the suggested approach in such situations has not yet been evaluated in a real-world container cloud scenario.
9.	S. Misra, P. V. Krishna, K.[9] Kalaiselvan	According to each user's QoS requirements, the suggested system can adaptively alter the resources allotted to them.	The methodology makes the assumption that all users' QoS requirements can be precisely described and assessed, which may not always be the case in actual use.

Sr.no.	Author(s)	Advantages	Disadvantages
10.	S. Sahoo, B. Sahoo and A. K. Turuk[10]	The algorithm is adaptive and can learn from past experiences to improve its performance over time.	In complicated cloud systems with plenty of tasks and resources, the proposed algorithm might not function as well.
11.	A. Valkanis, G. A. Beletsioti, P. Nicopolitidis, G. Papadimitriou and E. Varvarigos[11]	innovative method for optical network traffic prediction that may be of interest to academics in the area.	Since the proposed method is not thoroughly evaluated in the paper, it is challenging to compare its effectiveness to that of other approaches.
12.	Syed Q. Ali, Imthias Ahamed T. Parambath & Nazar H. Malik[12]	The suggested LA-based approach offers a dynamic and adaptable way to deal with shifting power demand.	In cases where the demand for power is particularly unexpected or varies quickly, the strategy could not be effective.unknowable.

Sr.no.	Author(s)	Advantages	Disadvantages
13	N. Rasouli, M. R. Meybodi and H. Morshedlou[13]	The suggested technique is able to dynamically adjust to system changes and eventually arrive at an ideal solution.	The algorithm relies on accurate workload and QoS information, which may not always be available or may be difficult to obtain in practice.

CHAPTER 3 :SYSTEM DEVELOPMENT

CloudSim

An open-source programme called CloudSim replicates cloud computing infrastructure and services. It is written in Java and was created by the CLOUDS Lab. It is beneficial to duplicate tests and results before developing software by modelling and simulating cloud computing environments. The system's architecture is built on CloudSim's use of Fog Nodes, which has facilitated the creation of cutting-edge applications with great scalability and low latency. The process of developing and maintaining applications has been revolutionised by CloudSim.

System Design

IoT applications that include several real-time operations frequently use cloud-fog computing since IoT devices are unable to process the enormous amounts of data produced by these apps. Traditional fog computing techniques, like virtual machines or Docker containers, are frequently utilised to overcome this constraint. Based on virtualization, which offers independence from hardware resources, these methods. In order to separate applications from the operating system, virtual machines are helpful. But they only employ one hypervisor, which could lead to a single point of failure. Virtual machines still provide benefits including portability, interoperability, quick boot-up times, and low resource needs. They also offer greater scalability than virtual machines, which makes them a desirable strategy in cloud-fog computing.

Virtual machines are focused on giving users more freedom and security, whereas containers are focused on the programme and its dependencies. Virtual machines are better suited for building a secure system, while containers are chosen for high availability and scalability. Depending on the needs of the user, either virtual machines or containers can be utilised for cloud-fog implementation. However, because virtual machines work well in constrained environments, this study makes use of them. The method makes use of virtual computers to supply computing resources, each with a different level of processing power. Although containers have benefits, the study opts for virtual machines because of their compatibility with the environment.

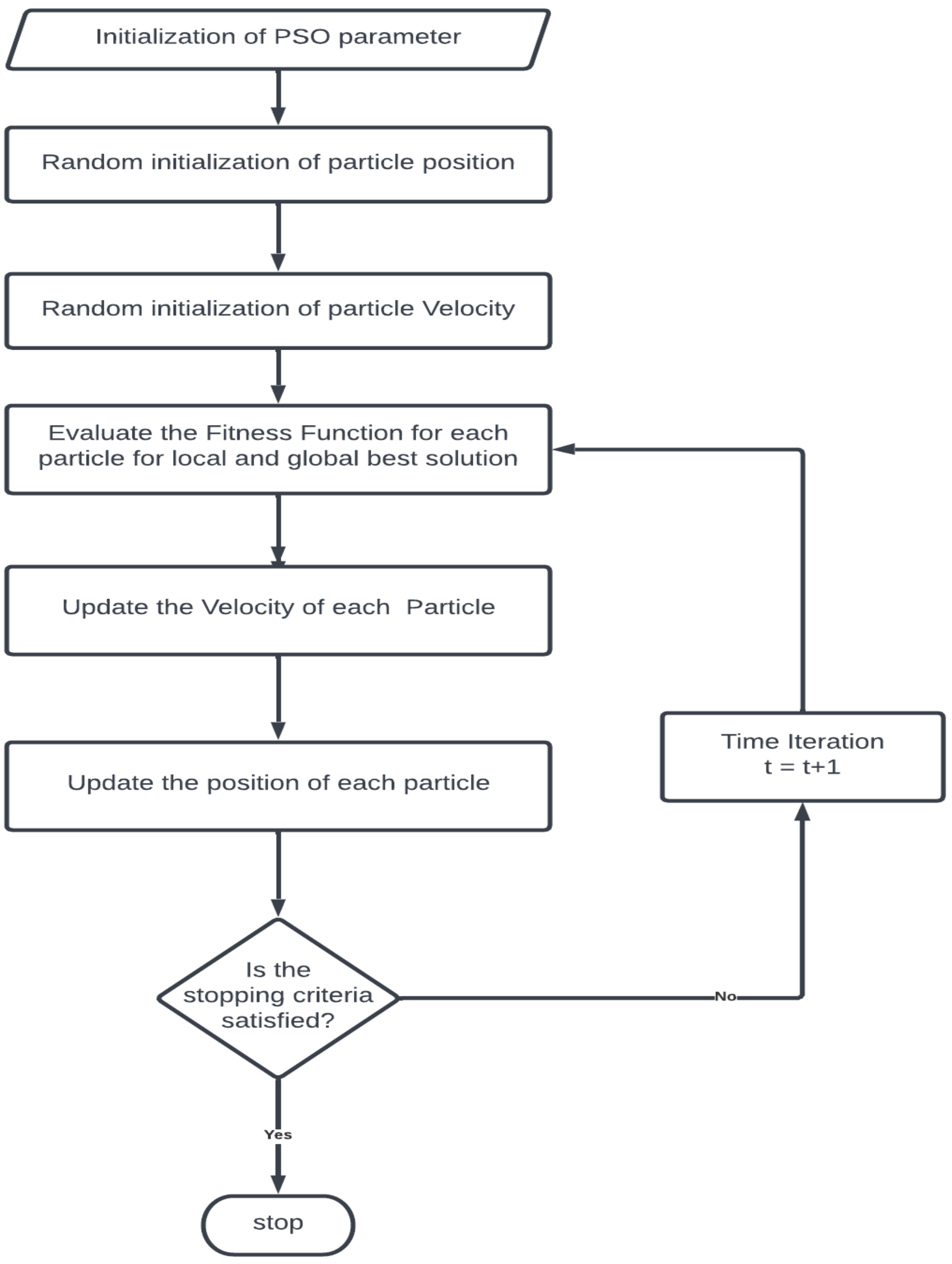
Scheduling Algorithm

The job scheduling methods covered in this section each have unique characteristics, advantages, and disadvantages. Some of the most significant and relevant algorithms are covered in the discussion that follows.

PSO - particle swarm optimization

This strategy is a population-based method that draws inspiration from the actions of flocks of birds, where the population is referred to as a swarm and the individuals inside it as particles. In the search space, each particle has a velocity that varies as a result of information exchange with other particles. Each particle contains a memory where the best individual particle positions and overall particle positions are stored for each iteration. Due to its allocation to a very small range, the ideal local position is kept as the ideal particle position. The ideal particle position overall is saved as the best global position. Each work is assigned to a machine that is available, which causes the partaicles to be instantiated at random intervals. The performance of the method can be enhanced by starting the PSO search with heuristic scheduling techniques like LJFP and MCT.

Particle swarm optimization task scheduler structure



Initialization of PSO parameter
Random initialization of particle position
Random initialization of particle Velocity
Evaluate the Fitness Function for each particle for local and global best solution
Update the Velocity of each Particle
Update the position of each particle
Is the stopping criteria satisfied?
No
Time Iteration t = t+1
Yes
stop

FIG 5. Particle swarm optimization SCHEDULER

Procedure for Particle Swarm Optimization (PSO) (as per the above fig)

The **initial step** defines the swarm size and the acceleration constant.

The initial position and velocity of each particle in the population are then generated at random in the **second stage**.

The final stage involves calculating each population solution's fitness value. The best options for the individual and the world are then assigned in **step four**.

The subsequent steps are continued in **step five** until the termination requirements are satisfied:

Step 5.1: With each repetition, each particle's position and speed are updated.

The population is assessed in **Step 5.2**, and the most effective individual and global solutions are updated.

Step 5.3: The process is looped until the termination conditions are met.

Finally, in **step six**, the results obtained so far are delivered.

Genetic Algorithm

The idea behind GA was to emulate natural system processes. By mixing genes and genetic operators, it employs a crossover function to produce new offspring, increasing variety by randomly altering the contents of people. GAs employ the selection operation, a probabilistic selection technique. Each solution in the population is evaluated by the algorithm, which then selects the top local and global solutions as the new best personalised and global solutions.

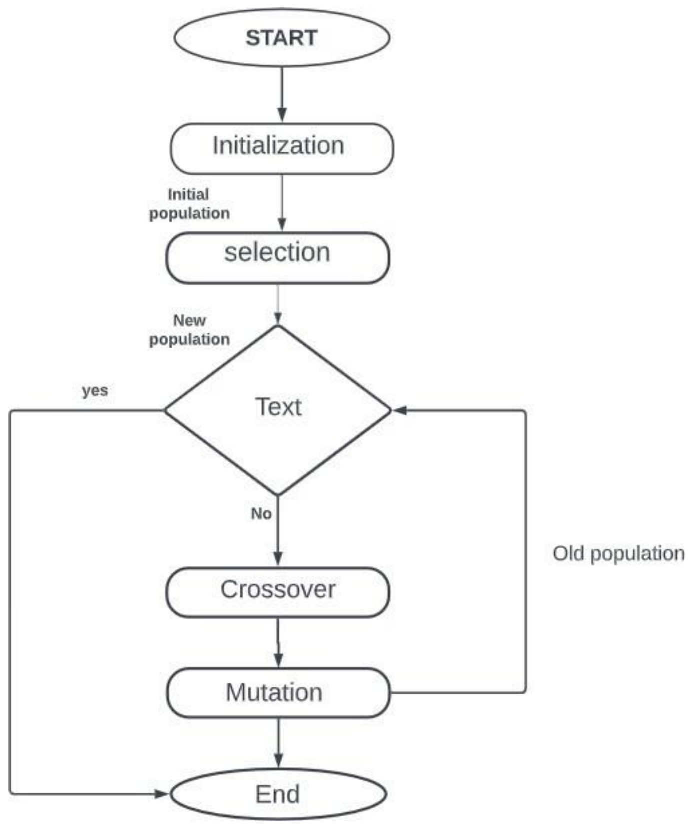


FIG 6. GENETIC ALGORITHM

Algo of Genetic Algorithm (GA) (As mentioned in Figure 6)

In order to put a genetic algorithm into practise, the following actions must be taken:

- Set the generation count to 0 at the beginning.
- Create a sample population at random.
- Assess each person's fitness function within the randomly created population.
  .
- Repeat the following steps until the termination criteria are met:
  - a. Increase the number of generations (t=t+1).
  - b. Use the selection operator to select a pivot point within the population.
  - c. Give each row a value (r) that was created at random.
  - d. Apply the crossover function to the chosen pairings if r is less than the pivot point.
  - e. Refresh the population.
  - f. Assign a random value (r1) to each gene in each individual.
  - g. Create a new random value for the selected gene within its domain to mutate the point.
  - h. Reassign the population.
  - i. Evaluate the fitness value of each individual.
- Until the termination requirements were satisfied, return the findings that were acquired.

Shortest Job First (SJF)

The process with the shortest execution time is chosen for execution next using the scheduling method known as "shortest task first". This tactic has two different preemptive and non-preemptive options. Due to its ease of use and capacity to shorten the time that other processes must wait for execution, it is regarded as ideal. "Shortest Job Next" (SJN) or "Shortest Process Next" (SPN), another comparable scheduling technique, chooses the process in the waiting queue with the least execution time for execution. SJN is a reactive algorithm rather than a predictive one. The preemptive SJN with the smallest remaining time is an enhanced variant of SJN. SJN is useful because it is straightforward and cuts down on the usual wait time for process execution. But it might result in process starvation for longer processes\

Algorithm for Shortest Job First (SJF)

- Add each process to the ready queue at the beginning.
- While the ready queue is not empty:

  a. From the ready queue, choose the process with the quickest execution time.

  b. Run the chosen process until it completes or is overridden by a quicker-running process.

  c. If the process completes, remove it from the system.

  d. If the process is preempted, put it back into the ready queue.
- End of algorithm.

Particle swarm optimization combined with a genetic algorithm.

The proposed Hybrid GA-PSO algorithm combines the strengths of both genetic algorithms (GA) and particle swarm optimization (PSO). GA is used for generating new solutions and maintaining diversity in the population, while PSO is used for exploiting the best solutions and searching in the local space. The algorithm starts by randomly initializing the population and evaluating their fitness values. Then, GA and PSO operators are applied iteratively to create new solutions and update the best personal and global positions. The algorithm terminates when the stopping criteria are met, and the best solution found so far is returned as the optimal task allocation. The hybrid approach of GA-PSO helps to overcome the limitations of each individual algorithm and achieve better performance in task scheduling problems.

The HPSOGA Algorithm

- The proposed HPSOGA method has a number of different parameters, including sample size, accelerator parameters, crossing rate, mutation chance, split number, split variables, split solutions, and iterations. These variables are essential to the optimisation process and can be changed to enhance the algorithm's performance.
- The counter variable t is initialised to 0 at the beginning of the procedure. Next, a population is created at random, and each response in the population is assessed.
    1. The algorithm employs the PSO technique to generate new solutions for the entire population.
    2. An intermediate population is selected from the present population in step 2 of the HPSOGA algorithm for task scheduling using the genetic algorithm (GA) selection operator. In order to provide better solutions in the following iteration, this stage assists in identifying the fittest members of the present population.
    3. In step three, the present population is divided into subpopulations, each of which consists of ideas for fixing each division. This is done to increase the search's variety and address the dimension problem .In this step of the HPSOGA algorithm applies an arithmetic crossover operator to each

subpopulation. This operator helps to combine the solutions from different subpopulations and create new solutions that inherit the characteristics of their parent solutions. The resulting offspring solutions are added to the overall population.

4. To avoid premature convergence, the HPSOGA algorithm now applies the genetic mutation operator to the entire population. This helps the population spread new genetic material and prevents it from stagnating in local optima.

- • In step four of the HPSOGA algorithm, each solution in the population is assessed for fitness by calculating its raw fitness value. The method then moves on to the following iteration after incrementing the counter t. Until the termination condition—typically a predetermined number of iterations or a satisfactory level of convergence—is satisfied, this process is repeated.

- The optimal solution is presented.

Learning Automata for resource scheduling

A probabilistic computational model that can learn and adjust to environmental changes is called Learning Automata (LA). In cloud computing systems, where resource availability and utilisation are continually changing, LA is a viable strategy for resource allocation. It has been discovered that using LA for resource scheduling works well for reducing task response times while ensuring resource efficiency.

A group of learning agents that interact with the environment and make decisions in response to feedback make up the LA-based resource allocation method. Based on the task's qualities and its current state, each agent, which represents a resource, decides whether to accept or refuse a task. The agents are given a set of parameters that can help them make better decisions as they gain experience.

Algorithm for Learning Automata

```
START

1. Set up the probability matrix to have a uniform distribution throughout the resources
available.
2. Randomise the reward values in the reward matrix to start with each resource-task
combination.
3. Clarify learning rate
4. For each task to be scheduled:
   a. Determine the resource with the highest reward value for the task
   b. Update probability matrix based on the chosen resource
5. Return final probabilities for scheduling

STOP
```

As a result, the algorithm's fundamental steps—choosing the best resource for each task based on the rewards from the reward matrix and updating the probability matrix to account for the best decisions over time—are summed up, these are some example in which we can use the LA algorithm

Resource Allocation in a Dynamic Way: Learning automata can be used to distribute resources in a dynamic way based on how the system's demands are changing. Learning automata, for instance, can be utilised in a cloud computing environment to allocate resources based on the workload and priority of various applications.

Load balancing: Learning automata can be used to distribute the system's workload across its various resources. For instance, learning automata can be used in a data centre to evenly distribute the load across many servers to guarantee optimum performance.

Using learning automata, it is possible to foresee when a resource will require maintenance or replacement. For instance, learning automata can be used in a production facility to anticipate when a machine needs maintenance based on usage trends.

CHAPTER 4 :PERFORMACE ANALYSIS

The purpose of this study is to assess the effectiveness of scheduling algorithm heuristics in a fog computing setting. The strategies are applied in a heterogeneous fog environment. The examination of the model's performance is based on the makespan time of the scheduling method..

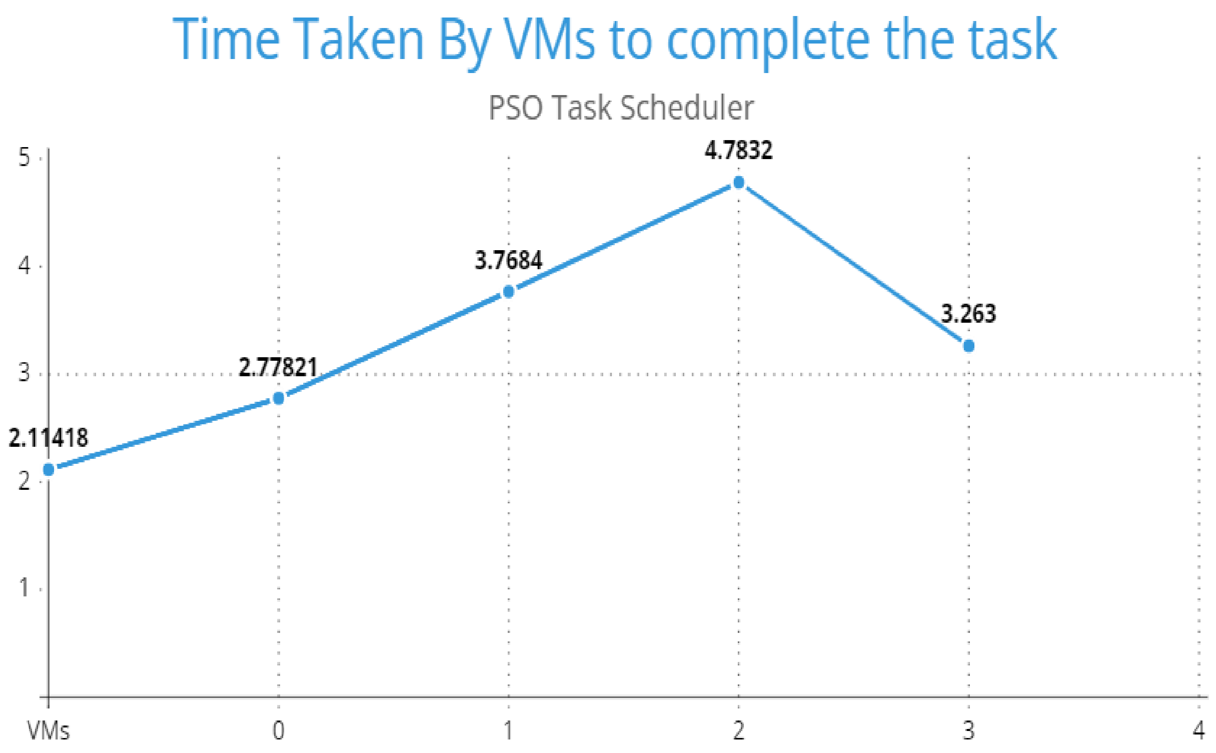
The following are the parameter constraints we took for the performance analysis of our model :

- Burst time is the length of time required for a process to finish under typical circumstances.
- Process arrival time is the moment a task is given to a processor to be completed.
- Makespan is the amount of time that passes between the beginning and end of a series of operations in a collection of machines.
- The completion time is the time when the process execution is complete.
- Turnaround time- This is the total amount of time spent at the processor, which is equivalent to the change between completion and arrival time.
- Waiting time is the interval between the time a process spends at the processor and the time it would normally take to finish. that is, the burst time minus the turnaround time.

Before using this type of scheduling approach, a processor must be informed of the burst time of the processes. This is also beneficial in batch processing, where waiting time is unimportant.

PSO Scheduler

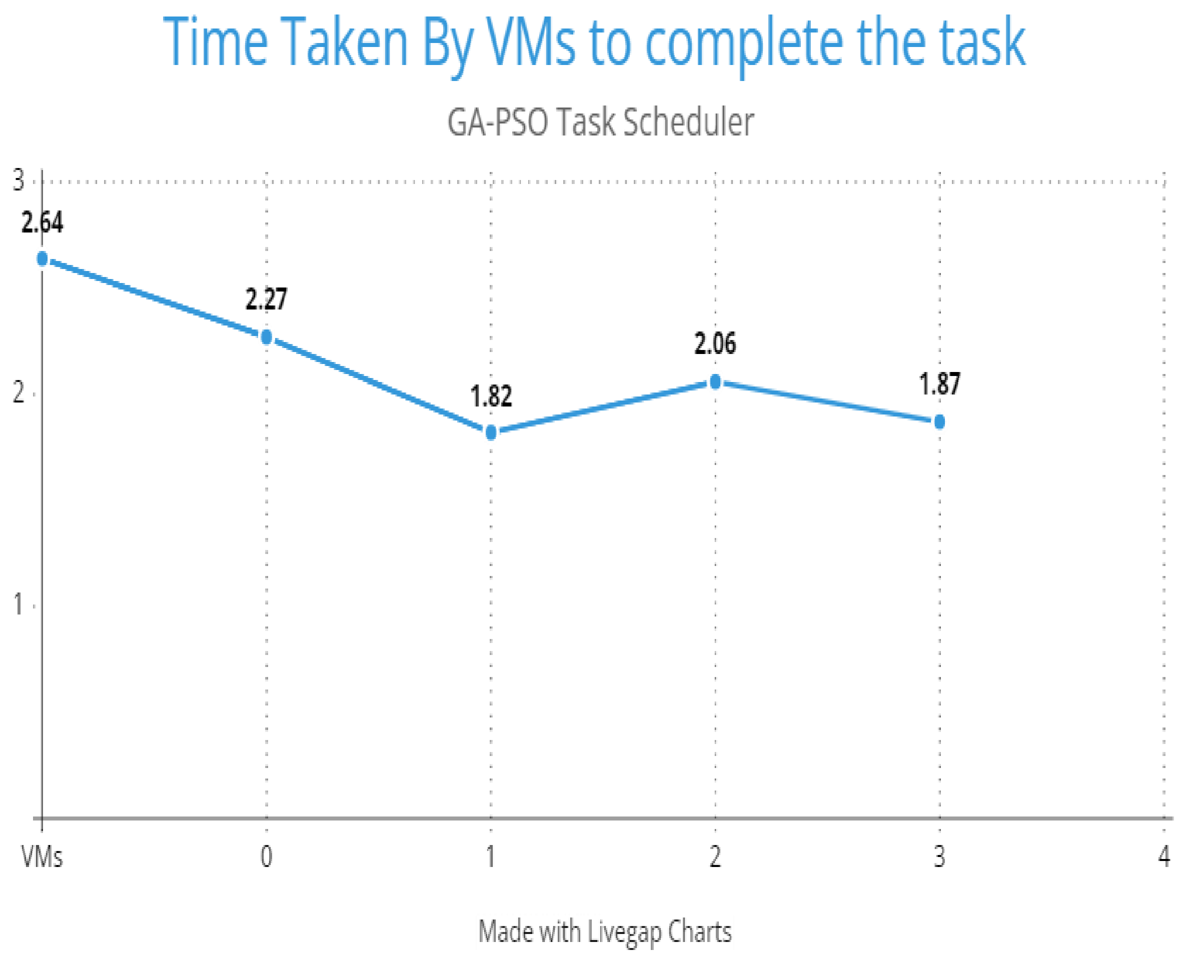
The start time, end time, and makespan time of the jobs were the three primary parameters used to analyse the PSO scheduler's performance. In a manner similar to the HPSOGA scheduler, the start time denotes the time at which a task is assigned to a resource, the end time denotes the time at which the task is finished, and the makespan time denotes the entire amount of time required to complete all tasks. The researchers were able to assess the PSO scheduler's efficiency and efficacy in terms of job scheduling and resource utilisation by looking at these metrics.



GRAPH 1. PSO VS VM ID

HPSOGA Scheduler

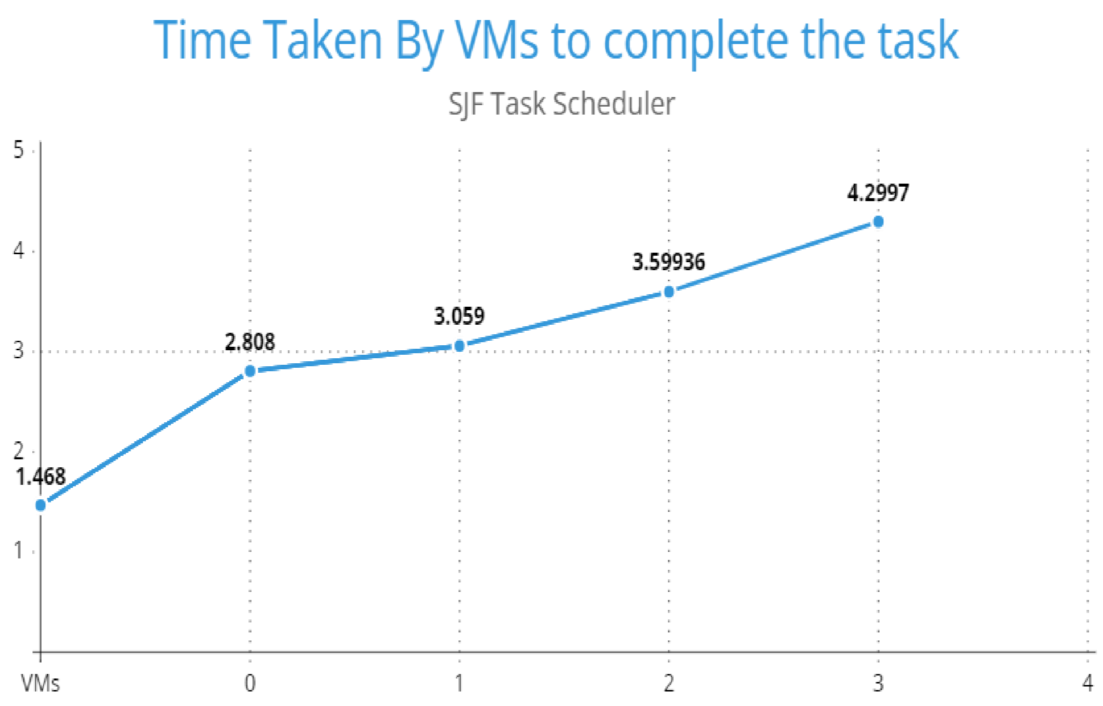
The start time, end time, and makespan time of the jobs were used as the basis for the performance analysis of the HPSOGA scheduler. The start time designates the moment at which a job is assigned to a resource, and the end time designates the moment at which the task is finished. The total amount of time needed to execute all jobs is called the makespan time. The researchers were able to assess the HPSOGA scheduler's efficiency and efficacy in terms of job scheduling and resource utilisation by looking at these characteristics.



GRAPH 2. VMs VS HPSOGA

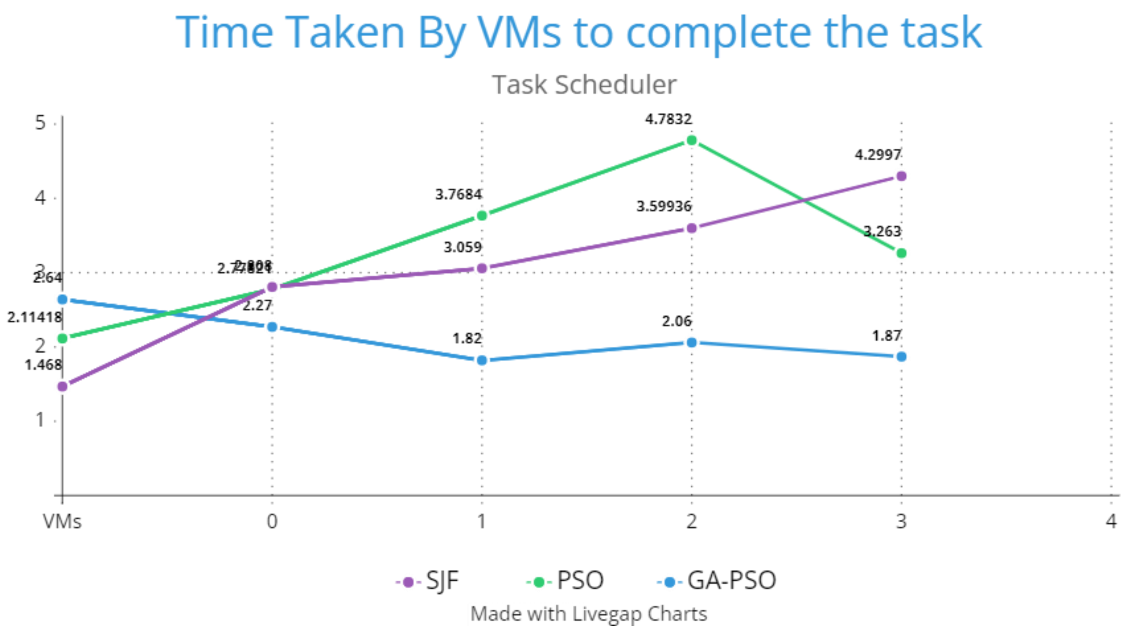
SJF Scheduler

The start time, end time, and makespan time of the jobs were the three main parameters used to evaluate the SJF scheduler's performance. The start time designates the moment at which a job is assigned to a resource, and the end time designates the moment at which the task is finished. The makespan time is the entire amount of time needed to finish all tasks, as well as the amount of time needed to finish the last job or task in an operation or process. The researchers were able to evaluate the SJF scheduler's efficiency and efficacy in terms of job scheduling and resource utilisation by looking at these criteria.



GRAPH 3. SJF VS VMs ID

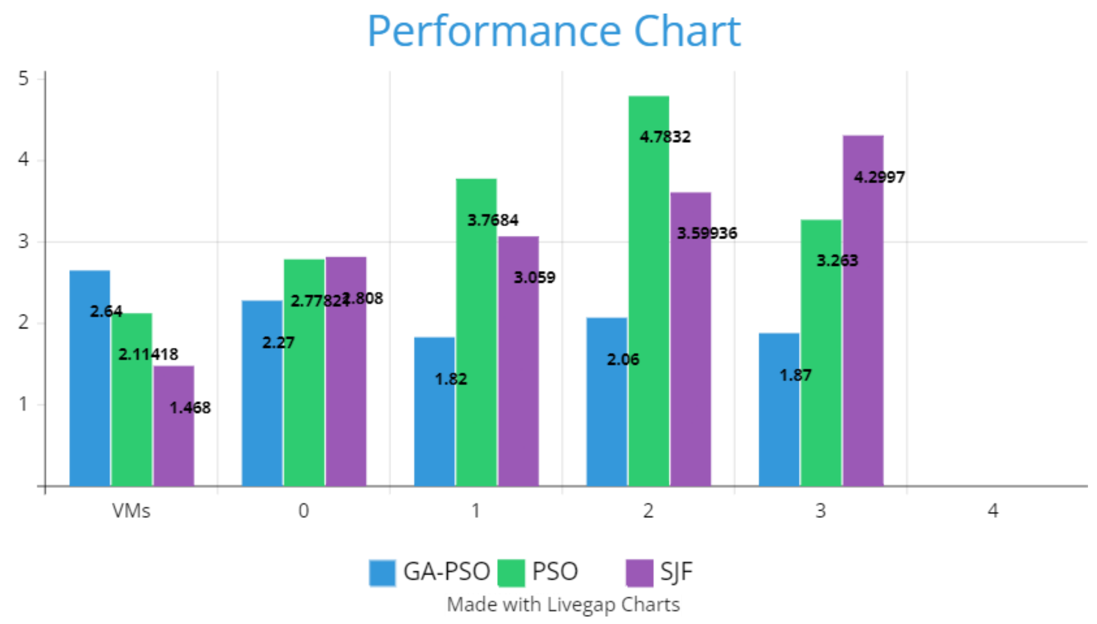
Performance Analysis



GRAPH 4. CHART FOR COMPARISION BETWEEN SJF, PSO AND GA-PSO

The graph shows that the performance of HGAPSO initially lags behind that of SJF and PSO algorithms, but as the number of tasks rises, it outperforms them.

COMPARISON OF MAKESPAN TIME OF SJF, PSO AND GA-PSO



GRAPH 5. CHART FOR COMPARISION BETWEEN SJF, PSO AND GA-PSO

Learning automata for resource scheduling analysis

Pseudo code of the implementation:

1. Start the probability matrix prob with a uniform distribution over the resources available.
2. Randomise the reward values in the reward matrix for each resource-task combination.

3. Explain learning rate alpha.

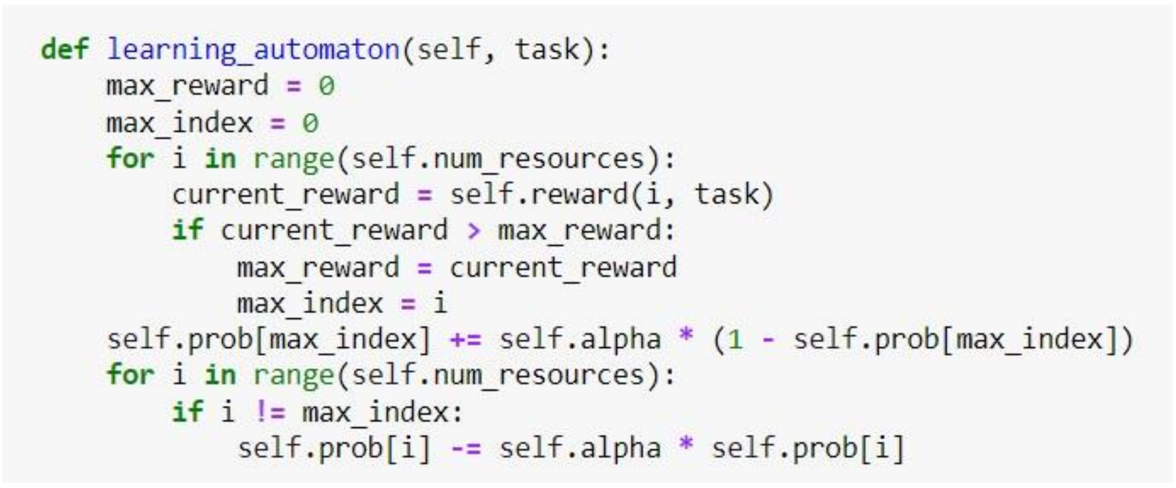
```
class CloudResourceScheduler:
   def__init__(num_resources, num_tasks, alpha):
      #  Initialize class variables
      self.num_resources = num_resources
      self.num_tasks = num_tasks
      self.alpha = alpha
      self.prob = initialise the probability matrix prob with a uniform
                    distribution using the  resources that are available.
      self.reward = # From the reward matrix, return the reward value for
                      the provided resource-task pair.
  def reward(resource, task):
         # Return reward value for a given resource-task pair from reward matrix
         return reward[resource][task]


   def learning_automaton(task):
# Create a new one-dimensional array to record the probability values for each resource in the system and update the probability matrix prob in accordance with the rewards each resource has earned for job p:
         r = reward(resource, task)
         if r is highest reward value for task:
            p[resource] = prob[resource] + alpha
         else:
            p[resource] = prob[resource] - alpha / (num_resources - 1)
      refresh  prob matrix with values in p array
```

```
    def schedule():
        # Execute the learning automaton algorithm for every task and return the final
probability matrix for every task in tasks:
            learning_automaton(task)
        return prob
```

4. Construct a CloudResourceScheduler object and fill it with the desired amount of resources, tasks, and learning rate.
5. Use the scheduling technique to get the final probability matrix for the object.
6. Print to the console the final probability for each resource.

Learning automaton method explanation:



```python
def learning_automaton(self, task):
    max_reward = 0
    max_index = 0
    for i in range(self.num_resources):
        current_reward = self.reward(i, task)
        if current_reward > max_reward:
            max_reward = current_reward
            max_index = i
    self.prob[max_index] += self.alpha * (1 - self.prob[max_index])
    for i in range(self.num_resources):
        if i != max_index:
            self.prob[i] -= self.alpha * self.prob[i]
```

The learning automaton algorithm is used for each task in the learning_automata approach. Here is a step-by-step breakdown of how this method works:

1. The method accepts a task parameter that denotes the action to be taken.

2. Based on the current state of the prob matrix, a new one-dimensional array p is made to contain the probability values for each resource.

3. A loop that iterates across all of the system's resources is begun. The reward method is used to retrieve the reward value from the reward_matrix for the current resource-task pair for each resource.

4. The resource with the highest reward value is then chosen by the algorithm as the most promising resource for carrying out the task.

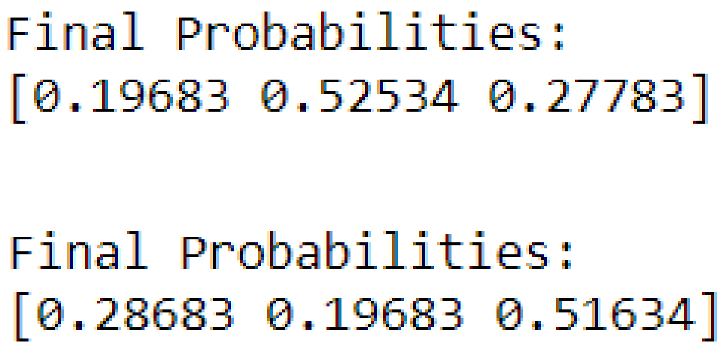
5. To make sure that the probabilities add up to 1, the probability value for the resource with the highest likelihood is increased by alpha (the learning rate), while the probability value for all other resources is decreased by alpha / (num_resources - 1).

6. The p array contains the revised probability values.

7.Based on the state of the p array, the prob matrix is finally updated with the current probabilities for each resource.

Results and analysis of the implementation

OUTPUT:



```
Final Probabilities:
[0.19683 0.52534 0.27783]

Final Probabilities:
[0.28683 0.19683 0.51634]
```

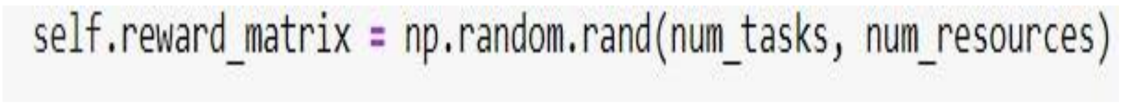
FIG.7: OUTPUT

The probability matrix that shows the likelihood of each resource being assigned to each task is the output of the learning automaton resource scheduling method. The likelihood that the relevant resource will be selected for the corresponding task is specifically represented by the value in each cell of the matrix.

The algorithm's particular use will determine the importance of the output. The probability matrix can be used to assign resources to activities in the context of cloud computing resource scheduling based on the most probable and effective combinations. The method is able to optimise system performance and guarantee that each task is carried out effectively by allocating resources based on the highest probability values.

The algorithm may also provide metrics like total resource utilisation, total execution time, and other performance indicators in addition to the probability matrix. These metrics can be used to assess an algorithm's performance and contrast various iterations or runs of the method.

Why random initialization of the reward matrix?!



```
self.reward_matrix = np.random.rand(num_tasks, num_resources)
```

The reward matrix must be randomly initialised in order for the algorithm to explore a variety of potential outcomes and avoid being trapped at a local optimum. The method is able to explore various resource-task pairings and converge to a solution that maximises the overall system performance by initialising the reward matrix with random values.

If the starting values of the reward matrix are inappropriate for the task and resource requirements, the algorithm may converge to a suboptimal solution if the matrix was initialised with fixed or predetermined values. The algorithm can explore various mixtures of resources and tasks and converge to a superior solution by randomly initialising the reward matrix.

It is crucial to remember that while the reward matrix's random initialization might lead to variations in the algorithm's outcomes, it is not the only factor that influences the algorithm's convergence behaviour and final solution. The performance of the algorithm can also be influenced by the learning rate parameter, task specifications, resource availability, and other variables. To guarantee the algorithm performs at its best, it is crucial to carefully select the beginning values and other algorithmic parameters.

ANALYSIS

The idea of a learning automaton, a mathematical representation of a decision-making agent that can learn from experience, serves as the foundation for the algorithm. The learning automaton depicts a scheduler in the context of resource scheduling for cloud computing that must choose which resource to allocate to each task based on previous performance and present task requirements.

Advantages:

Dynamic response to shifting circumstances: By revising the probability matrix in accordance with the algorithm may adapt dynamically to changes in the system, including the addition of new jobs or the removal of resources, by using the most recent reward values.

Scalability: The technique may be scaled to handle a huge volume of jobs and resources because it only needs to update the probability matrix for each task.

Flexibility: The algorithm is easily adaptable to various reward functions or decision criteria, making it appropriate for a range of application scenarios.

Disadvantages:

Initialization: The performance of the algorithm is significantly influenced by the initial values chosen for the probability and reward matrices. If the starting values are not selected carefully, the procedure could produce disappointing results.

Convergence rate: There is a chance that the algorithm's convergence rate will occasionally be delayed because it relies on making small-step modifications to the probability matrix in response to incentives.

Complexity: The method may be challenging to build and may require a lot of computing power to manage large systems with many jobs and resources..

CHAPTER 5: CONCLUSIONS

Fog and cloud resource integration has arisen as a critical component to meet the demands of IoT applications, generating profound changes in many parts of modern life. This paper introduces a meta-heuristic scheduling technique to efficiently plan jobs for IoT applications in a fog computing environment. By utilising complex algorithms, this technique can plan work in a dynamic and uncertain environment.

FUTURE SCOPE

1) Instead of using VMs, the algorithm can be improved by using containers which are expected to yield better results.
2) Containers are a more concrete option and have faster boot times compared to VMs.
3) The primary drawback of the algorithm is that it relies on static optimization techniques.
4) To make the algorithm more suitable for industrial applications, dynamic optimization techniques can be utilized.

REFERENCES

[1] Q.Liu, Y. Wei, S. Leng and Y. Chen, "Task scheduling in fog enabled Internet of Things for smart cities," 2017 IEEE 17th International Conference on Communication Technology (ICCT), 2017

[2] K. Benchikh and L. Louail, "Task scheduling approaches for fog computing," 2021 30th Wireless and Optical Communications Conference (WOCC), 2021

[3] H.S. Ali, R. R. Rout, P. Parimi and S. K. Das, "Real-Time Task Scheduling in Fog-Cloud Computing Framework for IoT Applications: A Fuzzy Logic based Approach," 2021 International Conference on COMmunication Systems & NETworkS (COMSNETS), 2021

[4] K. De Donno, K. Tange and N. Dragoni, "Foundations and evolution of modern computing paradigms: Cloud iot edge and fog", *IEEE Access*, vol. 7, pp. 150 936-150 948, 2019.

[5] T.S. Nikoui, A. Balador, A. M. Rahmani and Z. Bakhshi, "Cost-aware task scheduling in fog-cloud environment", *2020*

[6] *K*.D. Hossain, T. Sultana, V. Nguyen, T. D. Nguyen, L. N. Huynh, E.-N. Huh et al., "Fuzzy based collaborative task offloading scheme in the densely deployed small-cell networks with multi-access edge computing", *Applied Sciences*, vol. 10, no. 9, pp. 3115, 2020.

[7] R. R. Rout, S. Vemireddy, S. K. Raul and D. Somayajulu, "Fuzzy logic-based emergency vehicle routing: An iot system development for smart city applications", *Computers & Electrical Engineering*, kvol. 88, pp. 106839, 2020.

[8] L. Zhu, K. Huang, Y. Hu and X. Tai, "A Self-Adapting Task Scheduling Algorithm for Container Cloud Using Learning Automata," in IEEE Access, vol. 9, pp. 81236-81252, 2021, doi: 10.1109/ACCESS.2021.3078773.

[9] S. Misra, P. V. Krishna, K. Kalaiselvan, V. Saritha and M. S. Obaidat, "Learning Automata-Based QoS Framework for Cloud IaaS," in *IEEE Transactions on Network and Service Management*, vol. 11, no. 1, pp. 15-24, March 2014, doi: 10.1109/TNSM.2014.011614.130429.

[10] S. Sahoo, B. Sahoo and A. K. Turuk, "A Learning Automata-Based Scheduling for Deadline Sensitive Task in The Cloud," in IEEE Transactions on Services Computing, vol. 14, no. 6, pp. 1662-1674, 1 Nov.-Dec. 2021, doi: 10.1109/TSC.2019.2906870.

[11] A. Valkanis, G. A. Beletsioti, P. Nicopolitidis, G. Papadimitriou and E. Varvarigos, "Reinforcement Learning in Traffic Prediction of Core Optical Networks using Learning Automata," *2020 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI)*, Sharjah, United Arab Emirates, 2020, pp. 1-6, doi: 10.1109/CCCI49893.2020.9256655

[12] Syed Q. Ali, Imthias Ahamed T. Parambath & Nazar H. Malik (2013) Learning Automata Algorithms for Load Scheduling, Electric Power Components and Systems, 41:3, 286-303, DOI: 10.1080/15325008.2012.742943.

[13] N. Rasouli, M. R. Meybodi and H. Morshedlou, "Virtual machine placement in cloud systems using Learning Automata," *2013 13th Iranian Conference on Fuzzy Systems (IFSC)*, Qazvin, Iran, 2013, pp. 1-5, doi: 10.1109/IFSC.2013.6675616.