Efficient Resource Scheduling for Optimal Quality-of-Service in Fog Computing

A Thesis submitted to

UPES

For the award of

Doctor of Philosophy

in

Computer Science and Engineering

By

Gaurav Goel

May 2024

Supervisor

Dr. Shamik Tiwari

School of Computer Science

UPES, Dehradun, 248007: Uttarakhand, India.

Efficient Resource Scheduling for Optimal Quality-of-Service in Fog Computing

A Thesis submitted to

UPES

For the award of

Doctor of Philosophy

in

Computer Science and Engineering

By

Gaurav Goel

SAP Id: 500072316

May 2024

Supervisor

Dr. Shamik Tiwari

Professor, SCSE, IILM University

School of Computer Science

UPES, Dehradun, 248007: Uttarakhand, India.

DECLARATION

May 2024

DECLARATION

I hereby declare that the thesis entitled **"Efficient Resource Scheduling for Optimal Quality-of-Service in Fog Computing"** has been prepared by me, under the guidance of Dr. Shamik Tiwari, SCSE, IILM University, Gurugram. No part of this thesis has formed the basis for the award of any degree or fellowship previously.

GAURAV GOEL

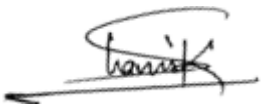
SAP Id :500072316

School of Computer Science

UPES, Dehradun, 248007: Uttarakhand

CERTIFICATE

I certify that Gaurav Goel (500072316) has prepared his thesis entitled "Efficient Resource Scheduling for Optimal Quality-of-Service in Fog Computing", for the award of PhD degree of the University of Petroleum & Energy Studies, under my guidance. He has carried out the work at the School of Computer Science, University of Petroleum & Energy Studies.



Dr. Shamik Tiwari
(Supervisor)
Ex-Professor (School of Computer Science, UPES Dehradun)
Currently, School of Computer Science & Engineering IILM University, Gurugram

ABSTRACT

The Cloud-Fog environment is excellent for providing clients with optimized services in their everyday regular tasks. The growth of IoT devices generates an exponential amount of data. To complete tasks on time, multiple service providers arrange the restricted resources available in the Fog computing environment employing optimization scheduling approaches. This study proposes an efficient hybrid optimization algorithms named Improved Grey Wolf Optimization algorithm(IGWOA),and Whale-Earthworm-optimization-algorithm (WEOA) for optimal resource management in a Cloud-Fog environment.It is difficult for independent algorithms to achieve a balance between exploration- and-exploitation. Too much exploration in the algorithm can lead to inefficiency and additional overhead, whereas much exploitation in the algorithms can lead to lost opportunities for improvement or inferior solutions. This work introduces an effective task allocation mechanism. This work improves the algorithm's exploration and exploitation stages to more effectively handle optimization problems in the Fog environment. It improves the exploitation phase to more effectively utilize found solutions and refines the exploration phase to more effectively probe the solution space. Together, these modifications strengthen the algorithm's capacity to operate in Fog-computing environments during both the exploration and exploitation phases.Simulations of the experiment are performed using the iFogSim toolkit, with appropriate environment parameters cost, makespan,response time, and throughput.Based on the selected parameters cost, makespan,response-time and throughput , the suggested strategy is compared to traditional optimization strategies.Extensive simulation demonstrates that the suggested technique outperforms the traditional scheme in terms of cost, time complexity, response time, and throughput.As a result, the suggested IGWOA,and WEOA approach is appropriate for IoT-Fog-Cloud environments.

ACKNOWLEDGMENTS

With a heart drenched in gratitude and eyes welled with emotion, I reflect upon a journey, not walking alone but alongside guardians who believed in me, especially during the times I doubted myself. My father and Mummy, my earliest mentors, have been the unwavering beacons in my life. Their silent sacrifices and invaluable lessons have been the bedrock upon which I built my aspirations. Their lessons of grit, resilience, and love have sculpted my character. To my beloved wife, who whispers encouragement in my most testing moments and whose embrace is my refuge. Every stride we've taken symbolizes our combined dreams and hopes. My Sisters, the reservoir of shared memories and dreams, has been my anchor and compass, constantly reminding me of our shared beginnings and boundless aspirations. A special tribute to my supervisor, whose immeasurable wisdom and nurturing guidance have not just shaped this work but also me as an individual. Their unwavering faith often felt like an invisible force, propelling me even when the path seemed treacherous. Their mentorship, much beyond academic guidance, has been a beacon of personal and professional growth. To my friends my extended family, for their constant chorus of support. Their camaraderie shared triumphs, and relentless belief in me have been the stars guiding my journey. This acknowledgment is a testament to a symphony of love, faith, and dedication from all who've touched my life. To each one of you, thank you for being the melody in this remarkable voyage.

Place: Ambala Cantt
Date: 03-05-2024 **Gaurav Goel**

Contents

List of Figures

List of Tables

LIST OF ABBREVIATIONS

AEO- Artificial Ecosystem Optimization
AEOSSA-Artificial-ecosystem-based-optimization with the Salp-swarm-algorithm
AWS-Amazon Web Services
BLA-Bees Life Optimization
BLEMO-Blacklist matrix-based-multi-objective
CDC-Count of dimension to change
CEA-Curie- commissariat à l'énergie ato- mique Curie
COA- Cuckoo optimization technique
CSDEO-cuckoo-search- differential Evaluation algorithm
CSO-Cat Swarm Optimization
CSP-Cloud service providers
CSPSO-Cuckoo-search-particle-swarm-optimization
DFGA- Double fitness genetic algorithm
EEOA-Electric-earthworm-optimization algorithm
EFT- Earliest finishing time
EHGB- Expand the Hungarian algorithm with a buffer queue
En-LoB-Energy and Load Balancing-Driven
FA-Firefly Algorithm
FIRMM-Fog-based IoT resource management model
FOA- Fruitfly optimization Algorithm
FOCAN-Fog-supported smart city network architecture
FRAS-Fuzzy-based real-time autoscaling
FSFC-First Come First Serve
GA-Genetic Algorithm
GABVMP- Genetic algorithm based Virtual machine Placement policy
GCP- Google Cloud Platform
GWO-Grey Wolf Optimization
HABBP- Hungarian algorithm for binding policy
h-DEWOA-hybrid-differential-evolution-enabled whale-optimization algorithm

HEFT- Heterogeneous earliest finishing time
HHA-Harris Hawks Algorithm
HMCR-Harmony Memory Consideration Rate
HPC2N-High Performance Computing Center North
HSA-Harmony Search Algorithm
IaaS- Infrastructure as a service
I-FA-Improved Firework Algorithm
I-FASC-Improved Firework Algorithm In Cloud-Fog
IGWOA-Improved Grey Wolf Optimization Algorithm
IoT-Internet of Things
LJFP-PSO-Longest job fastest Processor- Particle Swarm Optimization
MCT-PSO-Minimum Completion time-Particle Swarm Optimization
MGWO-Modified Grey Wolf Optimization
MIPS-Millions-of-instructions-per-second
MPSO-Modified particle swarm optimization
NSGA-2-Non-dominated Sorting Genetic Algorithm 2
PaaS - Platform as a Service
PSO-Particle Swarm Optimization
PWOA-Particle swarm based Whale optimization algorithm
QoS-Quality of Service
rank-ACO-rank Ant Colony Optimization
RIS-Resource information server
RSUs-Road Side Units
SaaS-Software as a Service
SJF-Shortest Job First
SLA-Service Level Agreement
SMP-Seeking Memory Pool
SONG-Swarm-Optimized Non-dominated sorting-Genetic algorithm
SPC-Self Positioning consideration
SRD-Seeking range of Dimension
SSA-Salp Swarm Algorithm
TCO-Total cost of ownership
TEC-Total Execution Cost
TET-Total Execution Time
VM-Virtual Machine
WEOA-Whale Earthworm Optimization Algorithm

Chapter 1

Introduction

1.1 IoT-Fog-Cloud Infrastructure

Fog computing, a distributed computing paradigm (Solutions, 2015), allows for actual-time data processing and Computation by extending cloud computing capabilities to the network's edge. By decentralizing processing resources, it tries to alleviate the latency, bandwidth, and scalability restrictions of cloud computing(Varshney & Singh, 2018; Aburukba, AliKarrar, Landolsi, & El Fakih, 2020; M. Bansal & Malik, 2020). For processing and storage, data is often transmitted to a distant data centre in traditional cloud computing. When dealing with real-time applications or massive amounts of data, in particular, this strategy may cause delays and require a significant amount of bandwidth. On the contrary, Fog computing entails the displacement of computational and storage assets in adjacent to the periphery of the system. (Naas, Lemarchand, Boukhobza, & Raipin, 2018), usually at edge devices or nearby data centres(Aslanpour, Gill, & Toosi, 2020; Hong & Varghese, 2018; Grover & Garimella, 2019). In Fog environment, the term **"Fog"** represents a metaphorical cloud that is positioned closer to the ground and represents the edge of the system, as illustrated in the Figure 1.1. Fog environment enhances speed of data processing, and reduces latency by strategically locating Fog nodes adjacent to the machines and detectors that generate the data(Ramzanpoor, Hosseini Shirvani, & Golsorkhtabaramiri, 2022a; Chhabra, Singh, & Kahlon, 2021; S. P. Singh, Sharma, & Kumar, 2020). The advantages of Fog computing are numerous. One primary benefit is the reduction in data volume that require to be broadcast to the cloud, resulting in less network crowd and diminished reliance on centralized data centers. Furthermore, it facilitates instantaneous and minimal delay data processing, rendering it

suitable for applications that require prompt responses or edge analysis.Thirdly, by keeping sensitive data near to its source, Fog computing improves data privacy and security.Fog computing can provide advantages to numerous industries, inclusive but not restricted to the industrial Internet of Things (IoT), healthcare, smart communities, and transportation. In these cases, local decision-making, edge analytics, and real-time control are made possible by Fog computing, which results in quicker response times, increased productivity, and improved user experiences.Broadly speaking, At the system periphery, Fog environment extends capabilities of cloud-computing (Mohammadzadeh, Chhabra, Mirjalili, & Faraji, 2024). Fog computing allows effective data processing, decreased latency, increased scalability, and increased reliability in a variety of applications by using scattered computer resources.

Fog environments consist of numerous Fog nodes situated in an environment with constrained storage and computation capabilities. These Fog nodes can be a Fog server, Fog device, and Fog gateway. Fog devices help store the data and Fog servers are helpful during computation. Fog gateways may help in transferring information or data between servers and devices. In networking routers, switches, and embedded servers may act as the Fog nodes. At present Fog computing is used by many application-like smart cities(T. Wang et al., 2019; Naranjo, Pooranian, Shojafar, Conti, & Buyya, 2019), smart industries, traffic management, smart cars, healthcare(Bharathi et al., 2020; Abdelmoneem, Benslimane, & Shaaban, 2020), etc. All these applications are real-time sensitive applications. Real-time data/information is required in these applications for the processing of tasks. For example, in the case of traffic management where automatic cars are plying, cars require all the real scenarios of traffic jams, humps, signals, etc. delay in the information may result in late applying of brakes and may lead to any serious accidents.

instead cloud infrastructure is located far away from IoT devices, which added bandwidth cost, energy consumption, and latency issues(Hameed, Jamil, & Ijaz, 2024; Naha, Garg, Chan, & Battula, 2020; Sing et al., 2022). on order to mitigate these problems, the Fog infrastructure is implemented on the intermediary surface that sits in the middle of the IoT and the Cloud layer. The IoT-Fog-Cloud infrastructure consists of three levels.The lowermost layer is the Internet-of-Things (IoT) layer, houses the devices and generates diverse forms of data from them. The uppermost layer functions as a cloud layer, housing high-capacity servers that offer storage and memory services. A layer of Fog is the intermediate level of infrastructure, comprising of Fog devices, Fog gateways, and servers with restricted capacity (Li, Liu, Wu, Lin, & Zhao, 2019; Goel &

Tiwari, 2023). It helps the fastest computation of tasks and helps in reducing latency, bandwidth, and energy consumption(N. Kaur et al., 2024; Gao et al., 2020; Abd Elaziz, Abualigah, & Attiya, 2021).

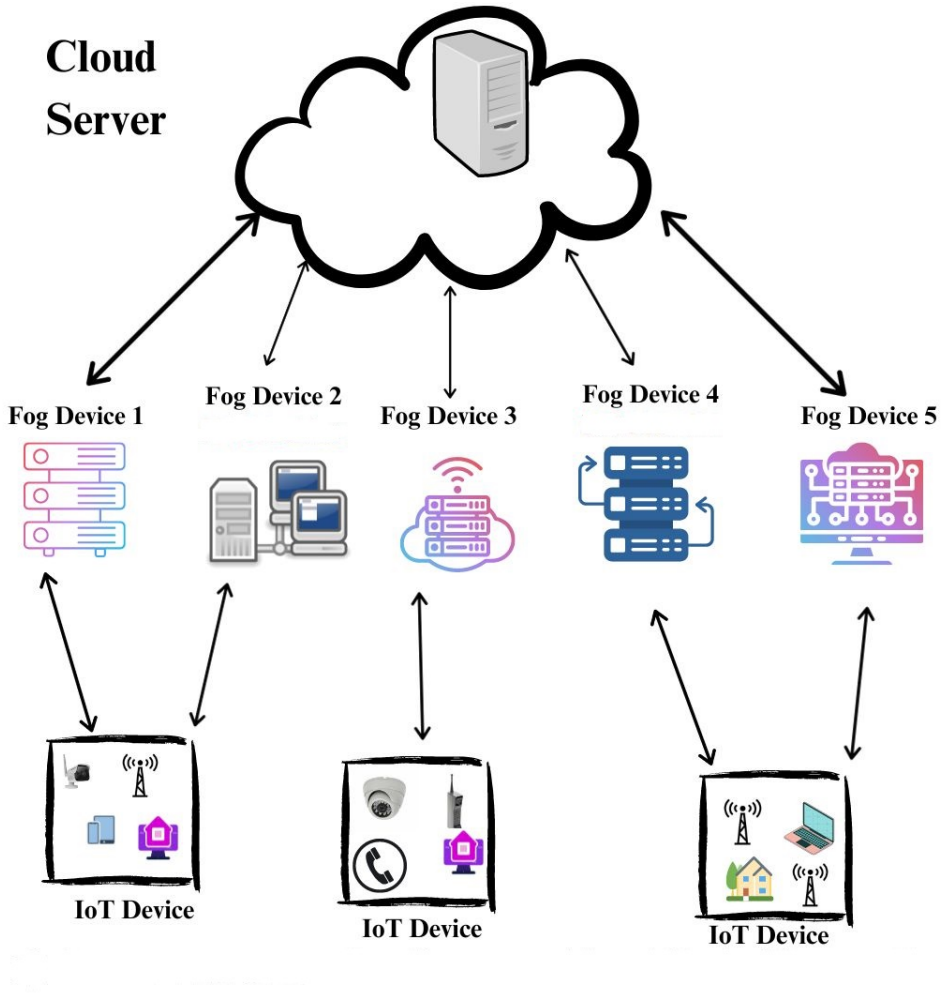


Figure 1.1: Traditional IoT-Fog-Cloud Architecture

A concept known as "cloud computing" mentions the providing of different computer services(Jeong, Baek, Park, Jeon, & Jeong, 2023; A. K. Singh, Swain, Saxena, & Lee, 2023; Agbaje, Ohwo, Ayanwola, Olufunmilola, et al., 2022) online. Cloud computing permit customers to remotely approach and utilize computing-resources via the internet, eliminating the requirement for a local server or personal computer to execute programs, store data, or perform other computing operations. These resources—which include storage, processing power, and a variety of services—are provided by independent companies referred to as cloud service providers (CSPs)(Saraswat & Tripathi, 2020; T. Alam, 2020; Anang, 2020). Cloud computing has emerged as a vital technology in today's digital landscape, allowing individuals and organizations to get access to

strong computer capabilities without incurring significant upfront hardware and infrastructure costs.

Here are a few crucial cloud computing features:(Ghosh & Grolinger, 2020; Krishnaveni, Sivamohan, Sridhar, & Prabakaran, 2021; Bai, Su, & Zhu, 2021)

I. **On-demand self-service:** Without contacting service providers directly, users can allocate computer resources as needed.

II. **Wide-ranging network access:**Standard devices including computers, tablets, mobile phones, and workstations can access network-based services.

III. **Resource pooling:** Cost savings and increased efficiency are possible when computing resources are combined to serve several users.

IV. **Quick elasticity:**it refers to the flexibility and agility that come from being able to quickly down or scale up computing resources in response to shifting needs.

V. **Measured service:** To enable cost optimization and resource management, cloud computing resources are measured, and customers are invoiced according to their utilization.

Because of its flexibility, affordability, and scalability, cloud computing has grown in popularity(Mohammed & Zeebaree, 2021; Waqar et al., 2023; Albarracín, Venkatesan, Torres, Yánez-Moretta, & Vargas, 2023). It has completely changed the way companies' function, allowing them to concentrate on their core skills, scale effectively, and innovate more quickly. Among the leading suppliers of cloud computing are GCP, Microsoft Azure, and AWS.

Generally, cloud computing services fall into one of three primary categories:(W. Wang, Du, Shan, Qin, & Wang, 2020; Hasan et al., 2024)

I. **Infrastructure-as-a-service (IaaS):** It issues virtual computing resources. Users can pay-per-use for networking infrastructure, storage, and virtual machines.

II. **Platform-as-a-Service (PaaS):** Giving a policy so users may design, execute, and organize applications without concerning about the problems inherent with architecture management. Development instruments, middle ware, and other services required for creating and deploying applications are usually included in PaaS packages.

III. **Software-as-a-Service (SaaS):** Offers web-based software applications in return for a subscription fee. Users can access programs via a web browser instead of installing or managing software locally.

An Internet of Things (IoT) device is any physical device with an internet connection that can send, receive, and collect data. Embedded sensors, software, and other technological components enable these devices to establish communication and interaction channels with other systems and devices via the internet(Jeong et al., 2023; Bell, 2024; Mahmudova, 2024). They can be anything from basic sensors to sophisticated machinery.
Smart homes, healthcare, manufacturing, transportation, and other sectors are just a few of the industries and uses for IoT devices. IoT device examples include:(Sabireen & Neelanarayanan, 2021; Safa'a et al., 2023; L. Xiao, Xu, Chen, & Chen, 2019)

I. **Smart thermostats:**With the help of a web interface or smartphone app, users can remotely monitor and manage a building's temperature with these devices.

II. **Wearable fitness trackers:**Health-related data acquisition devices include fitness bands and smartwatches, which monitor and record physical activity, sleep patterns, and pulse rate. Analyses of this information are possible via a computer or mobile device.

III. **Industrial sensors:** Industrial facilities have the potential to implement IoT sensors for the purpose of monitoring apparatus operations, detecting anomalies or issues, and optimizing workflows to enhance output and efficiency.

IV. **Smart appliances:**Appliances with internet of things capabilities, like washing machines, ovens, and refrigerators, can be operated from a distance and offer real-time information about their energy and status.

V. **Connected cars:**Contemporary automobiles are equipped with Internet of Things (IoT) technology, which enables features such as GPS navigation, instantaneous traffic notifications, remote diagnostics, and inter-vehicle communication.

VI. **Environmental monitoring devices:** IoT sensors can be used to track many environmental characteristics in metropolitan areas, industrial loca-

tions, and natural ecosystems, such as noise levels, air quality, and water quality.

Fog computing and Quality of Service (QoS)(Murtaza, Akhunzada, ul Islam, Boudjadar, & Buyya, 2020; Das & Inuwa, 2023; Ghobaei-Arani & Shahidinejad, 2022) are closely related. QoS techniques make sure that edge computing environments provide dependable, low-latency services while prioritizing vital applications and optimizing resource utilization. Fog computing QoS techniques improve user experience and allow new edge computing applications by efficiently managing resources, reducing latency, and boosting dependability and security.

1.2 Quality-of-Service in Fog Computing

In Fog computing, Quality of Service (QoS) describes the Fog infrastructure's capacity to satisfy performance standards and demands established by users or applications(Peng, Sun, Zhou, & Wang, 2023; Harnal, Sharma, Seth, & Mishra, 2022; Mani & Meenakshisundaram, 2020; Qu, Wang, Sun, Peng, & Li, 2020).In Fog environments, quality-of-service (QoS) is critical for the successful deployment of numerous applications, especially those that require real-time or low-latency processing.

Here are a few crucial QoS elements for Fog computing:(Vatanparvar & Al Faruque, 2018; Tuli et al., 2023; Hazra, Adhikari, Amgoth, & Srirama, 2021)

I. **Latency:**Fog computing processes data closer to the source, which lowers latency. To achieve their performance criteria, QoS mechanisms should make sure that applications that are sensitive to latency receive priority processing.

II. **Reliability:**To guarantee uninterrupted service availability, Fog nodes must be reliable. Reliability is preserved using efficient load balancing strategies, redundancy, and fault tolerance systems.

III. **Bandwidth:** The distribution of bandwidth across various services and apps operating on Fog nodes is managed via QoS methods. For critical applications to function properly, larger bandwidth guarantees could be necessary.

IV. **Scalability:** Environments for Fog computing should be scalable to meet changing needs for resources and workloads. To maintain performance

levels, QoS methods dynamically assign resources depending on changes in workload.

V. **Security:**Quality-of-service (QoS) considerations in Fog environment encompass safeguarding confidentiality, integrity, and security of data. Securing sensitive data and maintaining service quality require the implementation of critical security measures such as encryption, access control, and authentication.

VI. **Resource Management:**Effective resource management is necessary to maintain Quality-of-Service (QoS) in Fog environment. This involves the management of work allotment among Fog nodes, the maximizing of resource allotment, and the monitoring of resource utilization. (Songhorabadi, Rahimi, MoghadamFarid, & Kashani, 2023; Sabireen & Neelanarayanan, 2021; Ghobaei-Arani, Souri, Safara, & Norouzi, 2020)

VII. **Service Level Agreements (SLAs):**The expected QoS characteristics that service providers and users have agreed upon are defined by SLAs. To make sure that the level of service matches the required standards, QoS systems ought to enforce and monitor SLAs.(Zhao et al., 2021; Chang, Sidhu, Singh, & Sandhu, 2023)

VIII. **Orchestration and Scheduling:** Algorithms for orchestration and scheduling that take QoS into account are essential for maximizing resource use and achieving performance goals. These algorithms make intelligent placement and scheduling decisions by considering variables including workload characteristics, resource availability, and QoS needs.(Costa, Bachiega Jr, de Carvalho, & Araujo, 2022; Baranwal & Vidyarthi, 2021; Nazeri, Soltanaghaei, & Khorsand, 2024)

IX. **Adaptability:**Requirements for applications, resource availability, and network circumstances should all be considered by QoS methods. In dynamic Fog computing systems, QoS must be maintained by dynamic reconfiguration and optimization.

For the purpose of cater the diverse necessity of edge computing services and applications, fog computing necessitates the implementation of robust security measures, efficient resource management, judicious scheduling, and strict adherence to service level agreements (SLAs).

1.3 QoS Management Approaches in Fog Environment

Effective administration of quality-of-service(QoS) in Fog environment is a multifaceted undertaking that requires careful consideration of various aspects including user experience, performance, and dependability. Resource Scheduling, which allows Fog nodes to effectively share network, storage, and processing resources in response to changing application and service needs, is a crucial component of QoS management. Even in extremely dynamic and unexpected circumstances, this dynamic allocation guarantees optimal resource use and aids in satisfying quality of service (QoS) requirements(Malleswaran & Kasireddi, 2019; Moh & Moh, 2018; Li et al., 2019; Ayoubi, Ramezanpour, & Khorsand, 2021). Moreover, in Fog computing, QoS-aware task offloading(Hussein & Mousa, 2020; Aazam, Zeadally, & Harras, 2018) is essential for improving application performance and cutting latency. QoS-aware offloading systems optimize resource use and decrease response time, hence enhancing overall user experience. They do this by intelligently choosing whether to accomplish duties locally on edge devices, offload them to nearby Fog nodes, or process them in the cloud. In Fog computing, several QoS management techniques are used to accomplish this as shown in Figure 1.2 Now days, Resources Scheduling(Hameed

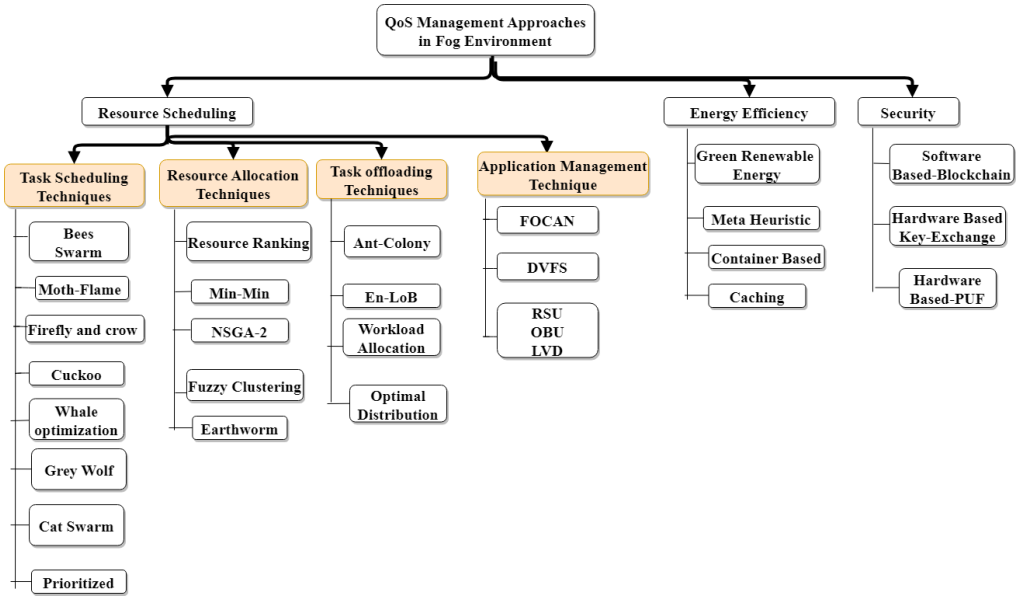


Figure 1.2: QoS Management Approaches In Fog Environment

et al., 2024; Varshney & Singh, 2018; Sun, Lin, & Xu, 2018) is a headache in the Fog environment due to the enlarge in the number of IoT devices. Most of the

population in the world is using smartphones, living in smart cities, and running smart cars. Due to these lifestyles' the resources of the Cloud-Fog environment are falling short.Due to the reality that these machines generate a substantial volume of data that requires storage and computation. There is a requirement of managing these data efficiently in the Fog environment. So that real-time data could be available. By offloading the task to the cloud someone can get the benefit of storage and memory, but the problem of delay, cost, and energy consumption is to be faced.Resource scheduling in Fog environment, conversely, has been subject of numerous studies. Nonetheless, in Fog environment, some continue to encounter challenges such as latency, bandwidth, and energy consumption.

Resource Scheduling(P. Wang et al., 2019; Al Ahmad, Patra, & Barik, 2020) may be managed through Task Scheduling(Goel, Tiwari, Anand, & Kumar, 2022), Resource Allocation(Goel & Tiwari, 2023), Task offloading(Kishor & Chakarbarty, 2021), and Application management(Ghobaei-Arani, Souri, Safara, & Norouzi, 2020; S. P. Singh, Kumar, & Sharma, 2020) techniques.To efficiently orchestrate resources across distributed edge environments and satisfy performance, scalability, and quality of service (QoS) needs, each of these strategies is essential. Let's examine each of methods in more detail as these are part of this research topic:

1.3.1 Task Scheduling

In Fog computing settings, where tasks must be efficiently completed across distributed edge devices and Fog nodes, task scheduling(Alizadeh, Khajehvand, Rahmani, & Akbari, 2020; Alsaidy, Abbood, & Sahib, 2022; Ghobaei-Arani, Souri, & Rahmanian, 2020) is an essential component. Task scheduling seeks to maximize system performance by strategically controlling the sequence and timing of task completion. Schedulers consider a number of things in order to accomplish this optimization:(Qu et al., 2020)

I. **Task Dependencies:** To make sure that dependent tasks are completed in the right order, task scheduling algorithms examine the dependencies between activities. In order to prevent inconsistencies or deadlock situations, tasks with inter dependencies may need to be scheduled either sequentially or concurrently depending on their relationships.

II. **Deadlines:** In order to meet application requirements or user expectations, tasks frequently have associated deadlines that must be met. Prioritizing

tasks according to their due dates helps task schedulers make sure that important tasks are completed on time to avoid delays or service interruptions.

III. **Resource Requirements:** Every task requires a unique set of resources, including but not limited to Central processor, memory, and access bandwidth. As per requirements of individual tasks and the available resources, task schedulers allocate resources among periphery devices and fog nodes. Optimizing the distribution of resources guarantees effective resource use and avoids resource conflict.

IV. **Reducing Idle Time:** By continually allocating jobs to available resources, task schedulers seek to reduce the amount of idle time on Fog nodes and edge devices. When there are no jobs that need to be completed or when scheduling is done inefficiently, resources sit idle. Schedulers enhance throughput and resource use by adding jobs to idle gaps.

V. **Optimizing Throughput:** By effectively employing the resources at hand, task scheduling algorithms aim to optimize the Fog computing system's total throughput. In order to do this, tasks must be scheduled to minimize processing bottlenecks, shorten task wait times, and maximize the use of computer resources.

VI. **Reducing Latency and Response Times:** Fog computing settings prioritize low-latency task execution to meet the real-time processing needs of Internet of Things applications, autonomous systems, and augmented reality. By allocating jobs to adjacent edge devices or Fog nodes with low network latency, task schedulers seek to reduce task execution times and communication latencies.

Moth-Flame(Ghobaei-Arani, Souri, Safara, & Norouzi, 2020), Cuckoo(Nazir et al., 2018), Bees-Swarm(Bitam, Zeadally, & Mellouk, 2018), Whale optimization (Ghobaei-Arani & Shahidinejad, 2022), Grey Wolf(Salimian, Ghobaei-Arani, & Shahidinejad, 2021), Cat-swarm(X. Xiao & Zhao, 2022),and Firefly Crow(Malleswaran & Kasireddi, 2019)etc.are traditional algorithms in Fog computing environments that maximize system performance, improve resource consumption, and guarantee timely task execution to satisfy application needs and user expectations by taking these elements into account. To fully benefit from Fog computing, including low-latency processing, scalability, and responsiveness in distributed edge contexts, effective task scheduling is necessary.

1.3.2 Resource allocation

A key component of efficiently managing resources(L. Yin, Luo, & Luo, 2018; Ni, Zhang, Jiang, Yan, & Yu, 2017) in Fog computing settings is resource allocation. It entails the purposeful distribution of networking, storage, and processing resources among jobs and applications according to their unique needs and priorities. A closer examination at resource distribution in Fog situations is provided here:

I. **Dynamic Provisioning:** Fog computing resource allocation is adaptable and dynamic, adapting in real time to shifting workload requirements and external circumstances. Resources are supplied or deprovisioned as needed to maintain maximum performance and scalability as workload varies.

II. **Optimizing Resource Utilization:**Algorithms for allocating resources are designed to optimize the use of networking, storage, and processing resources that are available in a Fog environment. These algorithms ensure that resources are utilized properly, reducing waste and increasing overall system efficiency by efficiently assigning resources to tasks and applications.

III. **Achieving Application-Specific Goals:** The objectives and specifications of the apps that operate in the Fog environment are what determine how resources are allocated. Applications with strict latency requirements, for instance, might use resources more carefully in order to reduce processing delays, whereas applications with high computational demands might need more resources in order to reach performance goals.

IV. **Balancing Objectives:** Allocation algorithms have to balance competing objectives, such as performance, energy efficiency, and cost-effectiveness. To guarantee the Fog environment operates sustainably and economically, for example, resource allocation strategies must take into account energy consumption and operational expenses in addition to performance maximization.

V. **Integration with Task Scheduling:** To guarantee that jobs are allocated to resources efficiently, resource allocation and task scheduling are tightly correlated. Decisions about task scheduling have an impact on resource allocation since different activities with varying demands and priorities may need different amounts of resources to be completed in order to

achieve their goals.

All things considered, resource allocation in Fog computing environments is essential for maximizing resource utilization, accomplishing application-specific objectives, and guaranteeing the Fog environment runs smoothly. Fog environments are able to satisfy performance, energy efficiency, and cost-effectiveness requirements by efficiently allocating resources to tasks and applications through the use of optimization algorithms, adaptive resource management approaches, and dynamic provisioning.Resource Ranking, Min-Min(S. Rehman et al., 2018), NSGA-2(Sun et al., 2018), Fuzzy Clustering(Li et al., 2019),and Earthworm (Kumar & Karri, 2023) etc. are some traditional techniques are provided by researchers for resource allocation.

1.3.3 Task offloading

In the Fog environment settings, task offloading is a key tactic that allows computationally demanding or latency-sensitive jobs to be transferred from edge machines to more powerful nodes on Fogs or servers on cloud (Aazam et al., 2018; Dzung, Tien, Tuyen, & Lee, 2015). This is an in-depth look into task offloading and why it matters in Fog environments:

I. **Offloading Criteria:** A number of criteria, including as the task's processing demands, the capabilities of the resources at hand, network conditions, and the criticality of latency, affect the choice to offload a task. jobs with strict latency requirements, jobs that require centralized processing, or tasks that require access to cloud-based resources are usually the ones for which offloading is taken into consideration.

II. **Latency Reduction:** Task offloading is a critical component for latency-sensitive applications as it significantly reduces processing latency and enhances responsiveness. By redistributing duties to resources that are in adjacent to the data origin or possess greater processing power, fog environments have the capability to minimize processing delays and deliver responses to users in real-time.

III. **Energy Conservation:** On edge machines with restricted resources, shifting workloads to Fog nodes or cloud servers can help save energy. In particular in mobile and IoT contexts, edge devices can run more efficiently and extend battery life by offloading tasks that demand considerable processing resources or high energy consumption.

IV. **Improved User Experience:** By lowering response times, guaranteeing constant service delivery, and boosting application performance, task offloading helps to improve the user experience overall. Fog environments can provide customers with high-quality services across a range of devices and network conditions by utilizing the extra processing power and storage capacity found in Fog nodes or the cloud.

V. **Dynamic Offloading Strategies:**Task offloading techniques can change depending on a number of dynamic variables, including resource availability, network congestion, and shifting workload patterns. Dynamic offloading algorithms are designed to continuously evaluate the state of system and create actual-time adjustments to offloading decisions in order to maximize energy efficiency, resource usage, and performance.

In Summary, in Fog computing environments, task offloading is an essential optimization strategy that facilitates effective resource usage, reduced latency, energy conservation, and improved user experiences. Fog environments can successfully exploit distributed resources to convince the request of a variety of applications and users by selectively offloading activities based on compute requirements, network conditions, and application priorities. Ant-colony(Hussein & Mousa, 2020), En-LoB(K. Kaur, Garg, Kaddoum, Gagnon, & Jayakody, 2019) etc. are traditional task offloading techniques are provided by authors.

1.3.4 Application management

In order to ensure that applications deployed across distributed edge devices and Fog nodes operate smoothly and perform at their best, application management (T. Wang et al., 2019; Mahmud, Srirama, Ramamohanarao, & Buyya, 2019; Das & Inuwa, 2023) is a crucial component of Fog computing environments. An extensive examination of application management and its essential elements is provided below:

I. **Automation of Deployment:** In Fog environments, application management entails automating the deployment of applications. Applications must be packaged, dependencies must be set up, and they must be smoothly deployed among edge devices and Fog nodes. Time-to-market for new services and updates is accelerated, human error reduction is achieved, and deployment procedures are streamlined through automation.

II. **Monitoring and Health Management:**Application management strate-

gies are employed to consistently monitor the health and performance parameters of applications. Monitoring tools collect data on response times, error rates, resource utilization, and other critical metrics with the purpose of assessing the functionality of an application and identifying any anomalies or snags in performance. Administrators may guarantee dependable service delivery, identify problems early, and troubleshoot difficulties effectively by proactively monitoring applications.

III. **Dynamic Scaling:** Workloads in Fog environments may vary as a result of shifting user demand, external factors, or application needs. Resource scaling for application management is done dynamically to account for workload variances. In order to achieve performance goals while maximizing resource utilization and cost-efficiency, scaling approaches include vertical scaling, which involves raising the resources allotted to an application, and horizontal scaling, which involves adding or removing instances of an application.

IV. **Fault Tolerance and Resilience:** Application management involves the implementation of fault tolerance and resilience techniques to guarantee continuous operation in the case of faults or disruptions. Fog applications are highly available and reliable because of techniques like replication, redundancy, and fail over procedures that assist lessen the effects of hardware failures, network outages, or software faults.

FOCAN(Naranjo et al., 2019), DVFS(Toor et al., 2019), etc. are some traditional techniques are used by researchers in their work. Fog computing environments may efficiently install, scale, monitor, and optimize applications to satisfy performance, reliability, and security needs by implementing comprehensive application management methods. By using application management strategies, administrators can keep Fog applications healthy and productive, guaranteeing smooth operation and providing users with high-quality services across distributed edge settings.

Concerning fog computing resource scheduling, this work addresses the subject. The purpose of this learning is to determine how effectively Fog computing allocates resources such as Central Processors, memory, and repository in order to achieve quality-of-service (QoS) metrics including response-time,makespan, and cost.

1.4 Resource Scheduling in Fog Environment

Resource scheduling in the Fog environments as depicted in Figure 1.3 is a challenging undertaking that necessitates innovative approaches to direct the distinct challenges present by dispersed computing at the system edge.Fog computing, in disparity to traditional cloud-centric paradigms, functions in a dynamic environment where a variety of devices with a range of capabilities interact in real time, requiring flexible resource management(Alizadeh et al., 2020; Alsaidy et al., 2022; Luo et al., 2019; Ogundoyin & Kamil, 2021) techniques. The focus on edge intelligence and autonomy in resource scheduling in Fog situations is one noteworthy feature. As Internet of Things (IoT) devices and sensor networks proliferate, Fog nodes frequently possess the ability to make decisions in order to evaluate local conditions and dynamically distribute resources accordingly. In addition to lowering need on central orchestrators, this decentralized strategy improves responsiveness and resilience to network outages.
Furthermore, data management and processing considerations are closely related to resource scheduling in Fog situations. The quantity of data created at the edge is lengthen exponentially, so effective data placement and processing solutions are critical. Resource schedulers have to use judgment when deciding whether to process data locally on edge machines, in adjacent Fog-nodes, or in the cloud layer in order to maximize bandwidth usage, data privacy, and latency while upholding application-specific specifications. Moreover, edge environments are dynamic, which means that algorithms for scheduling must be adjustable sufficient to modify to changes in network conditions, tasks, and device availability. These algorithms use machine learning, predictive analytics, and real-time telemetry data to proactively improve resource use, reduce performance bottlenecks, and forecast demand surges.
Resource scheduling(Mohammadi, Bahrani-Pour, Ebrahimi-Mood, & Farshi, 2024; C. Yin et al., 2024) in Fog environments frequently takes security, privacy, and regulatory compliance into account in addition to performance goals. The protection of sensitive data handled at the edge from tampering or illegal access makes it necessary to incorporate access control measures, encryption, and compliance checks into scheduling decisions.

Efficient distribution of compute, storage, and networking resources among dispersed Fog nodes is a crucial task in resource scheduling for computing environments. We aim to optimize resource utilization and reduce downtime while simultaneously satisfying Quality-of-Service (QoS) requirements for various ap-

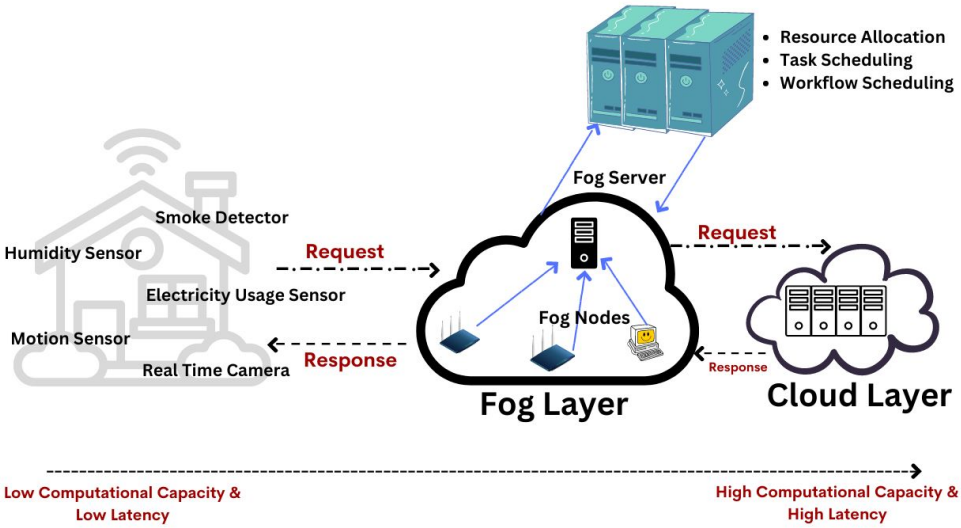


Figure 1.3: Resource Scheduling in Fog Environment

plications and utilities with this allocation.

Task offloading is a decision-making process that is fundamental to resource scheduling. Fog nodes must decide whether to handle a task in the cloud, offload it to a nearby Fog node, or execute it locally on edge devices. Numerous factors, including as task characteristics, resource availability(N. Kaur et al., 2024; Premalatha & Prakasam, 2024), network circumstances, and the unique QoS requirements of individual tasks, all have an impact on these selections. After jobs are transferred to Fog nodes, load balancing strategies are employed to disperse the workload among the available resources in an equitable manner. Load balancing algorithms dynamically balance the load and maintain acceptable performance levels for all applications and services by considering node capacity, workload characteristics, and QoS requirements.

Furthermore, Fog nodes dynamically distribute network, storage, and processing resources in response to shifting workload requirements and resource availability. Even in extremely dynamic contexts, dynamic resource allotment helps to satisfy quality-of-service (QoS)(K. D. Singh & Singh, 2024; Lv, Chen, Cheng, Qiu, & Li, 2024) objectives by optimizing resource utilization and reallocating resources when workload conditions change. It does this by ensuring that resources are allocated on-demand. Utilizing QoS-aware scheduling methods, resource scheduling also entails allocating computer resources optimally to various tasks according to their unique QoS requirements and priorities. These algorithms minimize reaction times and guarantee that vital jobs are processed first by taking into consideration variables including task deadlines, resource

availability, communication delays, and overall QoS objectives.
Furthermore, proactive resource allocation adjustments are made by using predictive resource management approaches, which foresee future workload demands. Fog environments are able to anticipate resource requirements and optimize resource scheduling decisions in advance to prevent performance deterioration and maintain constant service quality. This is achieved by utilizing machine learning models, predictive analytics, and historical workload data.
All things considered, resource scheduling in Fog environments is a difficult but necessary procedure that is critical to maximizing resource usage, reducing latency, and assuring that the quality-of-service (QoS) requirements of diverse applications and services are satisfied. Due to the efficacy of their scheduling of resources algorithms, Fog computing environments are capable of delivering dependable, responsive, and high-performing services to their users.

1.5 Fog Load Balancer

Due to inherent obstacles, cloud technologies have reached a level of maturity where they can give rise to the next generation of networks. The complexity of LAN and WAN technologies, particularly about payload and maintenance, was a driving force behind the conception of the cloud. It goes without saying that greater creativity was needed with traditional LAN and WAN technologies because of the proliferation of internet-related services(Baburao, Pavankumar, & Prabhu, 2023; Mutlag et al., 2023). The cloud technologies then made up for these improvisations. Initially, these cloud technologies assisted companies in lessening the burden associated with total cost of ownership (TCO). By dividing and specializing different types of company services, it has enhanced the services. Because of the rapid expansion of internet-related economics, enterprises are no longer needed to keep their own hardware inventory or worry about scaling up. These days, businesses may rent storage, bandwidth, memory, and networks, so they don't have to worry as much about growing when their company expands more quickly than the technology they purchased. As time went on, a huge number of cloud service providers appeared, and the industry shifted to focus more on services and subscriptions. Businesses could now provide a range of plans based on the payload and quality of service characteristics that were agreed upon with the clients. It has been felt recently that there are problems with the capabilities of cloud technology. This has to do with responsiveness and last-mile incorporation with cloud-services. It is also necessary for the sensors and last-stretch devices to have a continuous connection to the

cloud servers around-the-clock. Managing resources at the machine of cloud architecture is crucial for retaining cloud connectivity, particularly for remote weather stations located on mountains or wireless sensor networks positioned in the depths of the ocean. Synchronization with Fog devices and cloud servers is an imperative requirement for sensors or machines that cause time-sensitive information (Mattia, Pietrabissa, & Beraldi, 2023; M. Kaur & Aron, 2021). Data aggregators may be required in the final mile, connecting cloud servers and devices, as a result of their capacity to gather less sensitive data.Thus, the data payload needs to be handled based on the circumstances. Additionally, some sensors have separate timeframes and generate data asynchronously. Certain sensors contain biophysical data that is never acceptable to be tainted. Sensors continuously and intermittently produce data.

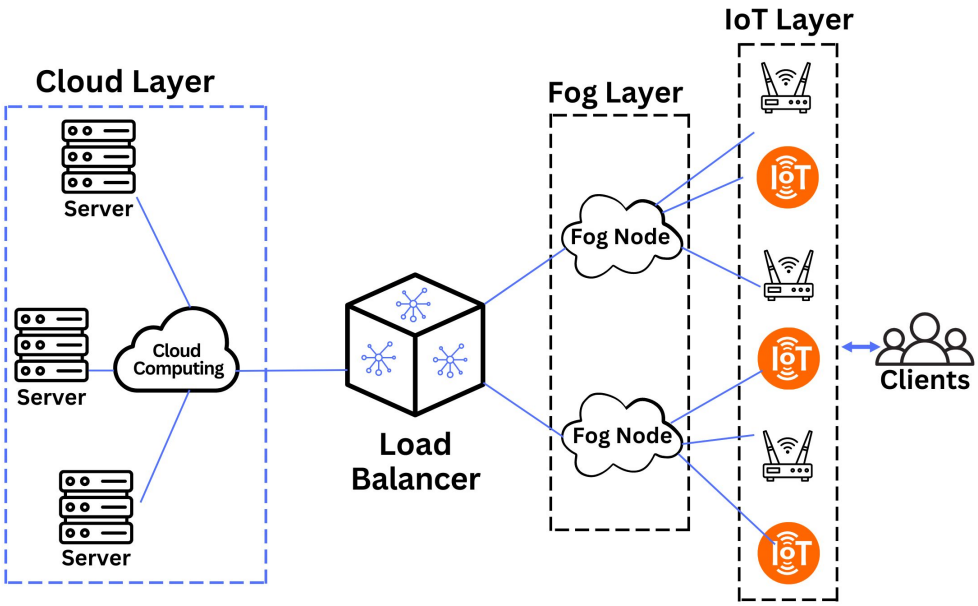


Figure 1.4: Load Balancer

The deployment position and application are the only discernible differences between a cloud device and a virtualized Fog device. A device is categorized as a Fog device if it is used to customize last-mile services or to sense data; otherwise, it is categorized as a cloud device(Talaat, Saraya, Saleh, Ali, & Ali, 2020; A. U. Rehman et al., 2020) as illustrated in Figure 1.4. The materials it is composed of could also be considered when classifying anything. The gadget may alternatively be classified as a Fog device if it has minimal computational

power and sensor capabilities. Algorithms that control the power and payload budgets as well as self-serving interfaces can be used to govern these Fog devices. These algorithms could be controlled from another device, or they might be stored on the virtual Fog device.

Power budgets in datacenters are calculated to either meet green requirements or lower operating costs. The job load directly affects the power budgets, and the limit of each is dynamically determined by the services' current demand. Above all, it needs to follow the guidelines for green technology. At the network's boundaries, the same ideas and guidelines are in effect. This serves as the fundamental basis for the IoT, a technological advancement that sanction the remote control of millions of processes. Remote access to these can be achieved by ensuring the presence of the necessary network infrastructure, which includes periphery, access, core, and processing center networking framework. Industry response indicates that software defined modules, mesh works, and smart algorithms will now drive scalability in terms of compute and resource management, instead of hardware expenditure. Thus, soft components will be used to assist in tasks like load balancing. Edge computing is required to deliver services without any transmission jitter, buffering, or latency. Evolution in innovation have led to the development of software-defined devices equipped with algorithms that generate a feedback loop. These devices aim to solve issues related to content retrieval time. Computation and data storage are distributed among peripheral machines and the cloud in the Fog computing scenario; therefore, load balancing is evaluative for maximizing resource fulfillment and ensuring efficient task accomplishment. The employment of virtual machine generators by load balancers is one of the creative tactics used in this situation.

In accordance with the workload and resource availability at any given time, these virtual machine generators are responsible for managing and dynamically providing virtual machines throughout the Fog network. The load balancer evaluates the status of the network, including the processing capability and proximity of available edge devices and cloud servers, when a new task or request reaches the Fog environment.

The load balancer makes an informed decision about where to instantiate additional virtual machines to manage the incoming workload based on this information. By placing virtual machine assembly on edge devices, which are in closer proximity to the data source or end consumers, it is possible to decrease latency and bandwidth consumption(Mutlag et al., 2023; Talaat, Ali, Saraya, & Saleh, 2022; Kashyap & Kumar, 2022). Alternatively, during periods of high demand or when there are limited resources available at the edge, the load balancer may

opt to route tasks to the cloud for faster processing.
This approach enhances the overall scalability and dependability of Fog computing environments, optimizes resource use, and enhances responsiveness by dynamically generating and controlling virtual machines based on fluctuating workload requirements. Moreover, it enables the Fog network to allocate resources in a flexible and adaptable manner, allowing it to promptly respond to dynamic workloads and emerging application needs.

1.6 Problem Statement

As per the literature review and gaps analysis done, there is a requirement of proposing an optimized resource scheduling algorithm for efficient resource management for applications in Fog environment, which should improve QoS. In most of the existing technique resource wastage occur due to the inability of dynamically removing and adding machine. It triggers the demand for an efficient optimization technique that dynamically removing idle machines, and adding machines whenever load on the system occurs. A mechanism is required that will avoid resource wastage and achieve QoS parameters such as throughput,Cost, makespan, and response time.

The proposed research work will formulate an efficient resource-scheduling scheme that can optimize QoS parameters such as throughput,Cost, makespan, and response time etc.

1.7 Gaps Analysis

After reviewing the literature,some gaps are identified in traditional techniques as follows:

1. **Communication Overhead Isn't Given Enough Attention:** In cloud-Fog systems, efficient task scheduling must account for communication overhead both between Fog nodes, and cloud server also between Fog nodes. However, some publications may not adequately address this issue or may employ overly simplistic assumptions that fail to take into account actual communication restrictions(Yadav, Tripathi, & Sharma, 2022),(Jamil et al., 2020),(Vijayalakshmi, Vasudevan, Kadry, & Laksh-

mana Kumar, 2020).

2. **Inadequate resource management:** Effective task scheduling should also include efficient resource management strategies like load balancing, fault tolerance, and scalability. In (Abu-Amssimir & Al-Haj, 2023), (Vijayalakshmi et al., 2020),(Abdel-Basset, Mohamed, Chakrabortty, & Ryan, 2021),(Abd Elaziz et al., 2021),(Tadakamalla & Menascé, 2021) works, resource management strategies are not fully utilized as per their scope. so a need of efficient load balancing, fault tolerant strategies is felt for effective resource management.

3. **Cost and energy efficiency trade-offs:**In job scheduling, finding the optimal equality allying cost and energy efficacy is challenging. so finding a stabilize allying cost and energy efficacy can be another optimization during task scheduling (Kumar & Karri, 2023), (Alzaqebah, Al-Sayyed, & Masadeh, 2019), (Ghobaei-Arani & Shahidinejad, 2022).

4. **Lack of variety in the workload:**A range of workloads and application types should be employed to assess the effectiveness and adaptability of task scheduling algorithms .However, some studies could focus on specific workload characteristics or fail to consider a range of application needs(Kumar & Karri, 2023),(Abu-Amssimir & Al-Haj, 2023),(Alzaqebah et al., 2019),(Dubey, Kumar, & Sharma, 2018).

1.8 Objective of Research

The objective of the current research problem is to design and implement a resource scheduling algorithm for a Fog environment to improve QoS parameters.

Sub-Objectives:

1. To implement peer competing scheduling and optimization techniques for Fog environment for enhancing QoS.

2. Design and implement novel and effective resource scheduling optimization techniques.

3. Verify and validate algorithms with peer competing techniques using some simulator tools like iFogSim, etc.

1.9 Contributions

1. A novel optimization algoriths are proposed to implement resource management efficiently in the Fog-Cloud environment.
2. Improved exploration and exploitation result in the best possible resource distribution.
3. Proposed and designed an autonomous "Load-balancer based task Allocation Frame-work" as per the three-layer architecture of Fog environment.
4. Validated proposed work with many experiments on performance metrics like cost, makespan, response-time, and throughput.

1.10 Thesis Layout

This thesis thoroughly examines the utilization of Fog network management algorithms for social causes, the plan of load balancing for Fog networks, and resource scheduling at the network's periphery. The Five chapters that make up the thesis are as follows:

- **Chapter1:Introduction:** The problem concept and research idea are covered in this chapter. The ideas, terminology, definitions, concepts,taxonomy, and procedures necessary to comprehend the specifics of this study are also covered in this chapter. This chapter also includes a discussion of the evolution of the many communication models needed to operate such arrangements.

- **Chapter2:Literature Review:** In this chapter description on a comprehensive summary of the research and related endeavors concerning resource scheduling in a fog environment is done. Here are Five segment in this division. The first segment inspects the traditional architecture and techniques related to Fog and cloud job scheduling. The chapter's second portion is devoted to strategies to handle latency in IoT-Fog-Cloud environment. The Third section describes the Virtual machine generation process in Fog environment. The forth chapter is discussion about smart cities applications.The previous chapter provides an explanation of how load balancing techniques can help with resource-management in a fog environment.

- **Chapter3:Framework and Methodology IGWOA for resource scheduling:**The elements pertaining to the Fog environment design are the main emphasis of this chapter. The chapter discusses the traditional Grey Wolf Optimization, Heterogeneous earliest finish time aspects. The chapter also describes proposed framework and methodology of IGWOA with Experimental evaluation and validation of technique.

- **Chapter4:Framework and Methodology WEOA for resource scheduling:**This chapter primarily emphasizes the elements associated with the design of the IoT-Fog-Cloud load balancer.The chapter covers how load balancer techniques is helpful for managing overload in the Fog environment. The segment talk about the traditional Whale-Optimization-algorithm,Earthworm optimization concepts.
  proposed structure and methodology of WEOA, along with the experimental assessment and technique validation, are also described in this chapter.

- **Chapter5:Conclusion and Future Direction:**This division gives an over--view and a conclusion for each step taken to accomplish the above specified goals. This chapter also emphasizes the groundbreaking advancements in the field of Fog environment.chapter also includes a segment that discusses societal consequences of the present investigation. final segment of chapter discusses methods and approaches that can be used to broaden the scope of this research and make valuable contributions to society.

Chapter 2

Literature Review

This segment outlines the investigation and related efforts pertaining to resource scheduling within the Fog environment. This segment comprises an examination and evaluation of related literature.

2.1 Job Scheduling in Fog System

An optimization approach based on Bees Life is proposed by Salim Bitam et al.(Bitam et al., 2018) to schedule jobs in the Fog environment. A exchange between allotted memory and Central processing unit execution time has been accomplished by researchers. Part of the Fog node computing methodology is the establishment of an administrative node. Five, ten, fifteen, and twenty Fog nodes make up each of the four types of Fog infrastructure that have been set up for simulation. Researchers have thought that one job can have several tasks, assuming that $Jtask_i$= $Jtask_1, Jtask_2, Jtask_3....Jtask_n$, for the optimization of the Bees Life method. Then $Jtask_{i1}^{5}$ tells, Fog node 5 implements task i1. C++ simulation was used to determine the Central processing unit implementation time and memory allocation parameters, and the cost function was then calculated using the results. Simulation findings indicate that BLA performs better than GA and PSO. Researchers have carried out a suggested method for static work scheduling. In the future, researchers intend to optimize bandwidth parameters in addition to putting the suggested strategy for movable Fog nodes into practice.

In their study, Shudong et al. (S. Wang, Zhao, & Pang, 2020) introduced a task scheduling perspective for a Fog environment that utilized an better firework

algorithm. By enhancing the firework algorithm, the researcher has presented a novel method to safeguard optimal solutions for task scheduling issues (I-FA). An improvised method (I-FASC) is provided for task scheduling. This technique divides jobs into three categories: bandwidth needed, computation-based storage requirements, and storage requirements. Task clustering and resource integration have been carried out based on task classification, and task allocation has been carried out in accordance with resource integration. A task controller is a mechanism that is set up in a system to assign tasks to resources based on categories. Researchers ran a simulation on the Alibaba cloud server to test load balancing and job processing time parameters. Algorithms rank-ACO, FSFC, and DFGA were compared in order to verify the outcome. Collate to the other techniques, the enhanced Firework algorithmic has demonstrated a faster processing time. The parameter for energy consumption has not been taken into consideration for evaluation.

Bushra et al.(Jamil et al., 2020) gives a job arrangement method for adjusting the Fog system's performance. In order to demonstrate how well Fog systems perform, researchers have worked with healthcare applications and implemented the SJF (shortest job first) scheduler technique. This class, which keeps track of completed and pending jobs in a queue, is derived from Cloudlet Scheduler. The foundation of the SJF scheduler technique is the ascending order of job sorting based on MIPS. Researchers have focused on decreasing application delays, optimizing the use of resources (CPU, RAM, and energy), and optimizing network utilization. They have also produced an algorithm for job scheduling. Researchers used the iFogSim toolbox for simulation. Comparing the simulation results to the FCFS technique, it is shown that delays are reduced by 32% and network utilization is reduced by 16%. Researchers have calculated the average CPU time used by each tuple to determine loop delay, and their findings demonstrate that the SJF technique improves delay by assigning greater priority to the most crucial loop. The idea of distributing loads to intermediary nodes has been applied to enhance Fog system network usage. The author of this study does not demonstrate an improvement in energy parameters when compared using the FCFS technique. In order to plan the capacity of devices with limited resources, researchers intend to apply analytical models, hyper- and meta-heuristics, and other strategies in the future.

A perspective for multi-objective task arrangement matters combining hybrid antlion-optimization and elite-based differential assessment was presented by Laith Abualigah et al.(Abualigah & Diabat, 2021). Scholars have employed authentic and artificial datasets to address issues related to job scheduling that

arise in the cloud environments. A multi-objective maximization perspective has been suggested to improve resource consumption while reducing reaction time, makespan, and imbalance (system load balance). The proposed MALO approach was collated to the particle swarm optimization,Genetic algorithm, and antlion-optimization-algorithm etc. so that assess and confirm the outcome, the Cloud Sim toolkit was utilized for this purpose. Scholars have established that the MALO method exhibits superior performance in phrase of parameter response-time, makespan, resource utilization, and degree-of-imbalance when collated to aforementioned algorithms. Furthermore, a statistical t-test has been conducted to verify the outcomes of the suggested methodology. Based on a comparison with the given algorithms, no significant improvement in makespan value was observed for a reduced number of tasks. Researchers intend to compare temporal complexity in the future by comparing it to the newest optimization techniques. They also intend to compare memory usage, peak demands, and overloads.

A method for thinking about the job scheduling issue in a cloud setting was put forth by Abdullah Alzaqebah et al.(Alzaqebah et al., 2019). A modified Grey Wolf Optimization (MGWO) method has been developed by researchers to resolve the work scheduling issue. so that the scheduler to effectively plan tasks for virtual machines (VMs), a resource information server (RIS) is essential. It gathers the most recent data on the resources that are available at VMs and provides it to the scheduler. Researchers have worked on the CloudSim toolkit's makespan, cost, and imbalance degree characteristics. The suggested algorithm is evaluated by comparing it to the GWO and Whale optimization algorithms. However, the results on one virtual machine show no improvement.

The subject of job/task scheduling in a cloud context has been inscribe by Seema A. Alsaidy et al.(Alsaidy et al., 2022). Using the heuristic algorithms LJFP and MCT, the authors have created a technique for randomly filling the search space in PSO. The hybrid algorithms MCT-PSO (Minimum-Completion-time-PSO) and LJFP-PSO (Longest-job-fastest-Processor-PSO) were proposed by researchers. TET(total-execution-time), makespan, TEC(total-energy-consumption) were the parameters used in a MATLAB algorithm evaluation. Two scenarios were compared with the aim of verify the results: the first involved changing the number of tasks while maintaining constant values for the variables makespan, TET, and degree of imbalance; the second scenario involved maintaining constant values for the parameters makespan, TET, and degree of imbalance while changing the number of VM. While MCT-PSO performs better than the other algorithms discussed, LJFP-PSO was unable to demonstrate performance in each situation.

In the future, researchers intend to load the search space using possibly another meta-heuristic algorithm in addition to any other heuristic methods.

An approach called moth flame optimization of job scheduling was presented by Mostafa Ghobaei-Arani et al.(Ghobaei-Arani, Souri, Safara, & Norouzi, 2020) for a cyberspace physical system in a Fog computing environment. Swarm tactics are also the foundation of the moth-Flame optimization methodology. Researchers have employed the idea of keeping a queue of tasks as they come in from devices, and then using the moth flame optimization algorithm to optimally distribute each task to Fog nodes. In order to demonstrate QoS in a Fog environment, researchers have taken task execution time and transfer time into consideration. To assess the outcome, NSGA-2, PSO, and BLA comparisons were made using the iFogSim toolkit simulation. The authors' comparison results demonstrate improvements over PSO, NSGA-2, and BLA in connection of job execution time,transfer time, and makespan. An inspection of a smart surveillance case study was conducted in order to validate the outcome. The outcome evaluation lacked certain important elements, such as cost, delay, and latency. In the future, researchers intend to integrate load balancing techniques with Moth Flame and to arrange tasks in the Fog environment using Grey-Wolf, and Whale-optimization techniques.

Dadmehr Rahbari et al. (Rahbari & Nickray, 2019)have demonstrated how to use scheduling algorithms based on the greedy knapsack problem to improve energy usage, execution cost, and sensor lifetime. Each object in a knapsack problem needs to have a weight and a profit. Thus, CPU usage and bandwidth have been regarded as two weights of the knapsack issue in the suggested approach. Similarly, the total amount of CPU time and bandwidth that the application module uses has been determined to be profit. In order to assess the outcome, simulation was carried out using the iFogSim toolkit in the DCNS and VRGame applications on energy and cost parameters. FCFS, delay priority, and concurrent algorithm comparisons were made. The suggested method performs better on parameters of cost energy and application loop latency than FCFS, delay priority, and concurrent algorithm. Future work on fault-tolerant and security in other applications is planned by researchers.

A unique method of work scheduling was put forth by R. Vijayalakshmi et al. (Vijayalakshmi et al., 2020) to maximize makespan and resource usage in Fog systems. In order to allocate resources to jobs efficiently, researchers have worked on the suffrage value (second minimum completion time-first minimum completion time) of each activity. The system receives a task execution time matrix as input for 30 tasks and 4 Fog nodes. Researchers utilize Java to compute

the results. In order to assess the outcome, the execution times of all four Fog nodes have been calculated, and the resource consumption and job assignment before and after scenarios have been compared and contrasted. work scheduling methods perform better than the conventional work scheduling scenario, according to result evaluation. Results for a very tiny platform with 30 tasks and 4 Fog nodes have been obtained by researchers. Researchers intend to work on grid and cloud systems in the future, as well as any application such as traffic congestion, healthcare, sanitation, and municipal services.

A hybrid technique combining the firefly and crow algorithms has been described by Senthil Kumar Avinashi Malleswaran et al. (Malleswaran & Kasireddi, 2019) for effective work scheduling in cloud environments. By employing the crow algorithm, researchers have improved the global optimization of the firefly method. In the suggested architecture, entities such as task handler, resource handler, and resource manager are useful for effectively distributing resources to tasks. The resource manager maintained track of all cloud-based resources that were accessible, the task handler handled tasks received from devices, and the resource handler handled resources to allocate resources to tasks effectively. Using the CloudSim toolkit, 30 virtual machines were set up, and the number of cloudlets was adjusted from 100 to 500. Researchers have compared the metrics of response time, completion time, and makespan between FF and CSA. The authors have demonstrated improvements over the previously mentioned methodologies in parameter completion time, makespan, and reaction time. However, the time it took to complete the tasks was nearly identical to the strategies described when there were 500 tasks.

in a Fog-cloud environment for IoT apps, Masoumeh Etemadi et al. (Etemadi, Ghobaei-Arani, & Shahidinejad, 2020) recommended a resource purveying method establish on an autonomous calculating model. A time sequence forecast model and a Bayesian instruction approach have been offered by researchers as a solution for the resource provisioning loop control technique known as MAPE-k (Monitor, Analyze, Plan, and Evaluate-knowledge based). Researchers employed the following methodology: the resource provisioning agent uses the MAPE-k loop control technique, the Fog community is encompass Fog nodes,Fog controllers, and admission control helps to pass requests on Fog or cloud based on deadline parameter. In order to appraise the results with respect to cost, CPU utilization, average number of Fog nodes allocated, latency violation, and authentic, smooth, and burst workloads, real and synthetic data sets were simulated using the iFogSim toolkit. The suggested method has been compared with DBN-GA (Deep Belief Network-Genetic Algorithm), QM

(Queuing model), FRAS, and E2DF for validate the outcome. When comparing the suggested method to the previously indicated procedures, a favorable comparison result was seen. The demonstrate method might have a significant temporal complexity. In the future, researchers intend to compare the suggested method of offloading the task with the methodology of offloading the task and to update the MAPE model's planning and analysis phase using a neural network prediction model.

Mohamed et al. (Abdel-Basset et al., 2021) enhanced Elitism-based Genetic Algorithm (IEGA), developed by researchers, is a method for resolving issues with scheduling of job/task in Fog environment. Algorithm includes two iterative improvement rounds. To provide more optimal solutions, the mutation rate and crossover are improved in the initial stage. To keep the technique from acquire persist in specific optima, more mutations are added to the optimal solutions that already exist in the second stage.

Mohamed Abd et al. (Abd Elaziz et al., 2021) Artificial Ecosystem-based-Optimization with Operator Salp Swarm Algorithm (AEOSSA), a revolutionary technology, has been introduced. To improve the explore for the best optimal solution within the search spaces, the algorithm integrates Operator Salp-Swarm-Algorithm(SSA) and Artificial Ecosystem Optimization (AEO). Utilizing MATLAB, experimental assessments were conducted with an emphasis on metrics of quality-of-service (QoS) including makespan and throughput.

Uma Tadakamalla et al. (Tadakamalla & Menascé, 2021)It was thoroughly explored the idea of offloading processing and transmission in IoT machines and the Fog-Cloud infrastructure. The analysis of response times and resource use when actions are carried out either on-premises or in the cloud was the main topic of discussion. A multi-class closed-form systematic line-up network model dubbed FogQN was developed to address this. An autonomic controller can use this model as a starting point to dynamically switch between analyzing Fog and cloud information.

Fatma M. Talaat et al. (Talaat, 2022) For allocation of resource in IoT devices, the authors devised a method known as EPRAM (Effective-Prediction-and-Resource-Allocation-Methodology). Allocation of resources,Data pre-processing, and an efficient prediction module were the three key areas of study for EPRAM. The authors specifically suggested a plan for the healthcare system that intended to foretell instances of heart attacks. To achieve this, a Probabilistic Neural Network (PNN) for heart attack detection was trained using a training dataset. The suggested technique also included a deep reinforcement learning algorithm for allocation of resources.

Table 2.1: Comparison Table of significant literature techniques for Job Scheduling

Author name and Reference	**Type of Algorithm**	**Technique used**	**Evaluation tool**	**Performance Metric**	**Pros**	**Cons**
Salim Bitam (Bitam et al., 2018)	Meta Heuristic	Bees Life	C++	• CPU execution time • Allocated memory • Cost	Cost reduced.	Static job scheduling performed.
Shudong Wang (S. Wang et al., 2020)	Meta Heuristic	Extended Firework	Alibaba cloud server	• Processing time of tasks • Load balancing	Cluster-based classification of tasks and resources.	Cost and Energy consumption parameters are missing.
Bushra Jamil (Jamil et al., 2020)	Heuristic	Shortest Job first	iFogSim	• Delay • Network usage	Reduced Delay.	Mature Heuristic technique is used for scheduling.
Laith Abualigah (Abualigah & Diabat, 2021)	Meta Heuristic	Modified Antlion	CloudSim	• Response time • Makespan • Degree of imbalance.	Effective Response Time.	Time Complexity is high.

Table 2.1: Comparison Table of significant literature techniques for Job Scheduling

Author name and Reference	**Type of Algorithm**	**Technique used**	**Evaluation tool**	**Performance Metric**	**Pros**	**Cons**
Seema A. Alsaidy (Alsaidy et al., 2022)	Hybrid	Longest job fastest Processor-PSO, Minimum Completion time-PSO.	MATLAB	• Total execution time • Makespan • Total energy consumption • Degree of imbalance	Improvement in makespan.	Longest job fastest Processor-PSO unable to show performance.
Mostafa Ghobaei-Arani (Ghobaei-Arani, Souri, Safara, & Norouzi, 2020)	Meta Heuristic	Moth flame optimization	iFogSim	• Task execution time • Makespan	An efficient algorithm for reducing task execution time.	Load Balancing is missing.
Dadmehr (Rahbari & Nickray, 2019)	Scheduling Algorithm	Greedy knapsack	iFogSim	• Delay • Cost • Energy	Cost reduction.	Fault-tolerant needs to be considered.

Table 2.1: Comparison Table of significant literature techniques for Job Scheduling

Author name and Reference	**Type of Algorithm**	**Technique used**	**Evaluation tool**	**Performance Metric**	**Pros**	**Cons**
R. Vijayalakshmi (Vijayalakshmi et al., 2020)	Scheduling Algorithm	suffrage value	Java	• Makespan • Resource utilization	Resource utilization and task assignment are considered effective.	Worked on a very small platform.
Senthil Kumar Avinashi Malleswaran (Malleswaran & Kasireddi, 2019)	Hybrid	Firefly and crow algorithm	CloudSim	• Response time • Makespan • completion time	Efficiently handles resource scheduling.	Makespan has not improved as compared.
Masoumeh Etemadi (Etemadi et al., 2020)	Resource provisioning technique	Time series prediction model and Bayesian-learning technique.	iFogSim	• Cost • Delay	Resource provisioning is done with the MAPE-k control loop constructively.	Time complexity is high.

Table 2.1: Comparison Table of significant literature techniques for Job Scheduling

Author name and Reference	**Type of Algorithm**	**Technique used**	**Evaluation tool**	**Performance Metric**	**Pros**	**Cons**
Mohamed Abdel-Basset (Abdel-Basset et al., 2021)	Meta Heuristic.	Improved elitism-based genetic algorithm	Java	• Makespan • Flow time • Energy consumption	Helps to deal with the problem of misplacing good solutions.	The technique is not performed for dependent task scheduling. Requirement of more makespan improvement.
Mohamed Abd Elaziz (Abd Elaziz et al., 2021)	Meta Heuristic.	Artificial ecosystem-based optimization with operator salp swarm algorithm	MATLAB R2018b	• Makespan • Throughput	Successfully outperforms AEO, PSO, HHA, SSA, and FA.	Techniques may be tested on more objectives like cost, and energy consumption.

Table 2.1: Comparison Table of significant literature techniques for Job Scheduling

Author name and Reference	**Type of Algorithm**	**Technique used**	**Evaluation tool**	**Performance Metric**	**Pros**	**Cons**
Uma Tadakamalla (Tadakamalla & Menascé, 2021)	NA	Multi-class closed-form systematic line-up network model FogQN	JMT tool	• Response time • Cost utilization.	This model is utilized by an autonomic controller to change the processing between Fog and cloud dynamically.	Still, improvement is required in this work on designated parameters.
F. M. Talaat (Talaat, 2022)	Meta Heuristic	EPRAM	iFogSim	• Data preprocessing • Resource allocation • effective prediction	Successfully outperform other traditional algorithms for QoS parameters.	Distributed behavior of technique is required for more efficiency.

2.2 Delay Sensitive applications

In order to decrease latency, Raafat O. et al. (Aburukba et al., 2020) give the idea for scheduling the requests made by IoT devices in Fog environment. A genetic algorithmic has been used by researchers to work on the suggested strategy. First, Lingo software was used to simulate the results for the occupant area and max-

imum number of repetitions for GA parameters in relation to the total runtime latency. The demands that were made and the resources that were available vary in these comparisons. Next, Branch and Bounding of Lingo technique is used to test the genetic algorithmic efficacy. When comparing the contrast between problem size and latency, the Lingo algorithm and GA have produced identical results. In the end, a round robin was used for the parameters of latency time and completion time, and a comparison and contrast between GA and priority tight queuing was waited for for fair queuing. The latency has improved with the suggested method by 21.9% to 46.6%. Similarly, the suggested method has seen a 31% improvement in request deadlines. Later, researchers expect to use a different heuristic algorithm to work on multi-objective functions.

The method of scheduling of resource in a Fog environment environment using NSGA-2 (a non-dominated-sorting-based genetic-algorithm) has been described by Yan Sun et al. (Sun et al., 2018). Fog node clusters have been employed for this objective. Migration of tasks can be carried out both within and between Fog nodes within a cluster. Researchers have proposed NSGA-2 to optimize QoS in Fog nodes within the same cluster. The entities responsible for assigning Fog nodes for task computation and monitoring resources for allocation reasons were the Fog resource supplier and manager nodes. Researchers have designed the fitness function, generated a non-dominant set, improvised crowding distance computation, and encoded and initialized chromosomes and population for the NSGA-2 algorithm. A suggested method for MATLAB parameters that lowers service latency and boosts the stability of full job execution has been put into practice by researchers. The suggested method (RSS-IN) has demonstrated superior outcomes when compared to FIRMM and Random on the specified parameters by altering the quantity of tasks and resources. This was done in order to verify the outcome. The cost parameter was absent from the suggested method for demonstrating algorithm optimization.

In the Fog environment, Hina Rafique et al.(Rafique et al., 2019) give out the effective resource management strategy NBIHA. Scholars have employed the Modified PSO technique for resource management, as well as the Modified Cat Swarm optimization and Task Management techniques. The on-demand methodology has been applied by researchers to task and resource management. Resources will be assigned to jobs in accordance with demand, and for any resources that remain, the two best values (bestfitres1 and bestfitres2) must be determined. Resources will be assigned to jobs if another work requires them and the demand for those resources fits bestfitres1. If not, bestfitres2 will be used to verify demand. In order to find a memory pool larger than bestfitres1

and bestfitres2, MCSO was utilized. The average response time, resource utilization, energy consumption, execution time, and cost were all simulated using the iFogSim toolbox. The comparison was carried out using the MPSO, SJF, and FCFS optimization algorithms in order to confirm the outcome. In terms of execution time, and cost etc. NBIHA implement superior than all algorithmic listed. The energy usage of the lesser number of Fog devices, such as 5 and 10, did not improve in this way. Researchers intend to apply a reinforcement-learning method for resource management in the future.

Santhosh Kumar et al. (Kumar & Karri, 2023), In an attempt to enhance Quality of Service (QoS), the researchers put out a hybrid electric-earthworm-optimization-algorithm to handle the issue of effectively slate the jobs in a Fog environment. The method combines the interest of the earthworm and electric-fish-optimization algorithms, both of which help to beneath system delay and energy consumption. The suggested method makes use of both active and passive electrolocation to update positions, increasing the effectiveness of the scheduling technique.

Yadav et al.(Yadav et al., 2022), To improve the makespan and cost in optimization tasks, the authors presented a amend class of the fireworks algorithm. To boost the firework algorithm's performance, they used opposition-based learning and differential assessment methods. Taking into account both phases of the algorithm ensured that it would not become mired in local optima, thereby accelerating its convergence.

Abu-Amssimir et al. (Abu-Amssimir & Al-Haj, 2023), For latency-sensitive IoT applications, the researchers introduced a greedy-edge-placement technique that focuses on minimizing delay. The suggested method seeks to reduce latency and increase throughput effectively.The placement stage and the application selection stage with the greedy delay minimization algorithm are the two phases of the proposed methodology.

Ogundoyin et al.(Ogundoyin & Kamil, 2023), The authors presented a mixed optimization technique that coalesce the Firefly technique with PSO. The study's main objectives were to address problems with sojourn rate and trust, energy use, and node capacity in a particular system. The researchers used a method known as liner-sum-weight to handle the combination of distinct objective functions.

Hussain et al. (Hussain et al., 2023),For the purpose of precisely address the problems of latency and energy-sympathetic automotive applications, the article described a novel technique for vehicular Fog computing. The focus was on data offloading from automobile gadgets to base stations and roadside units (RSUs). To optimize the system, the authors suggested solving a multi-impartial ob-

stacle problem known as a Swarm-Optimized-Non-dominated-sorting-Genetic-algorithm (SONG).

Amit Kishor et al. (Kishor & Chakarbarty, 2021) An IoT-Fog-Cloud system's latency problem was addressed using the Smart Ant colony optimization (SACO) technique. In order to achieve the lowest possible latency, the authors carefully evaluated the distribution and computing of tasks within the Fog environment.

Yaser et al. (Ramzanpoor et al., 2022a) A many-objective cuckoo- search-optimization approach was given by the authors in order to address the effective application deployment in the Fog infrastructure. The resolution of problems with resource usage, bandwidth utilization, increased power consumption, and subpar Quality of Service (QoS) levels was the main focus of the article.

Mostafa et al. (Ghobaei-Arani & Shahidinejad, 2022) At Fog environment, the researchers have suggested a framework for autonomous resource management that makes use towards MAPE (Monitor, Analyse, Plan, Execute) method. utilization about whale-optimization-algorithmic for the effective arrangement of IoT services for cloud, Fog, and server nodes has also been covered. Throughput and energy usage are taken into account by the framework as objective functions for resource allocation in IoT services.

Heena Wadhwa et al. (Wadhwa & Aron, 2022) The IoT-Fog-Cloud system's resource allocation difficulty was discussed in the study. The authors explored the intricacy of the process and emphasised the challenges that come with resource allocation in the present cloud environment. The researchers implemented a zero-hour approach to address this problem, in which higher priority jobs are given a priority level of zero during resource allocation. The objectives of this policy were to enhance task performance and optimize allocation of resources.

Table 2.2: Comparison Table of significant literature techniques for Delay Sensitive applications

Author name and Reference	**Type of Algorithm**	**Technique used**	**Evaluation tool**	**Performance Metric**	**Pros**	**Cons**
Raafat O. Aburukba (Aburukba et al., 2020)	Meta Heuristic	Genetic	Lingo	• Latency • Completion time	Reduced Latency.	Works on a single objective function.
Yan Sun (Sun et al., 2018)	Hybrid	Non-dominated sorting-based genetic algorithm (NSGA-2)	MATLAB	• Latency	Improvement in latency is seen.	The cost parameter is not considered.
Hina Rafique (Rafique et al., 2019)	Meta Heuristic	Modified Cat Swarm	iFogSim	• Average response time • Energy consumption • Execution time • Cost	Acceptable improvement is seen in Average response time and energy consumption.	Performance degrades for a smaller number of Fog devices.

Table 2.2: Comparison Table of significant literature techniques for Delay Sensitive applications

Author name and Reference	**Type of Algorithm**	**Technique used**	**Evaluation tool**	**Performance Metric**	**Pros**	**Cons**
Santhosh Kumar et al. (Kumar & Karri, 2023)	Meta-heuristic	A hybrid electric earth-worm optimization algorithm	CloudSim	• Cost • Makespan • Energy Consumption • Execution-Time	successfully outperform traditional techniques on used performance metrics.	offloading mechanism and workload management task is missing in this work.
Yadav et al. (Yadav et al., 2022)	Meta-heuristic	Modified Fireworks algorithm	iFogSim	• Cost • Makespan	significance of the proposed approach is verified on used performance metrics.	There is still a need for improvement in the makespan metric.
Abu-Amssimir et al. (Abu-Amssimir & Al-Haj, 2023)	Meta-heuristic	Greedy-edge-placement technique	iFogSim	• Delay • Throughput	Successfully provides high-quality services by reducing the network bandwidth, latency, and consumption of energy.	Response time needs to be calculated for showing efficiency in the network.

Table 2.2: Comparison Table of significant literature techniques for Delay Sensitive applications

Author name and Reference	**Type of Algorithm**	**Technique used**	**Evaluation tool**	**Performance Metric**	**Pros**	**Cons**
Ogundoyin et al. (Ogundoyin & Kamil, 2023)	Meta-heuristic	Hybrid Particle swarm and Firefly algorithm	MobFogSim	• Energy-Consumption	The proposed technique is successfully outperforming traditional techniques.	Cost and makespan may be part of performance metrics.
Hussain et al. (Hussain et al., 2023)	Heuristic	vehicular based Fog computing	PYOMO, VANET	• Delay • Energy-Consumption	Proposed approach provides improved quality over NSGA-2 and SMPSO.	Some more improvement is required in this work for considered metrics
Amit Kishor (Kishor & Chakarbarty, 2021)	Meta Heuristic	Smart Ant colony optimization algorithm	MATLAB	• Latency	SACO outperforms all mentioned traditional techniques RR, MPSO, and Bees life.	Power consumption, makespan, and throuhput also should be considered in this work.

Table 2.2: Comparison Table of significant literature techniques for Delay Sensitive applications

Author name and Reference	**Type of Algorithm**	**Technique used**	**Evaluation tool**	**Performance Metric**	**Pros**	**Cons**
Yaser Ramzanpoor (Ramzanpoor et al., 2022a)	Meta Heuristic	cuckoo search optimization algorithm	iFogSim	• Bandwidth usage • Higher power consumption.	proposed technique outperforms all the mentioned traditional techniques.	Resource requests should be known in advance.
Mostafa Ghobaei-Arani (Ghobaei-Arani & Shahidinejad, 2022)	Meta Heuristic.	MAPE technique and whale optimization technique.	iFog Sim	• Throughput • Energy consumption	Lessen the service delay and consumption of energy.	A small improvement is noticed in energy consumption when compared with the PSO technique.
Heena Wadhwa (Wadhwa & Aron, 2022)	Meta Heuristic	Zero-hour policy	iFogSim	• Cost • Bandwidth	proposed approach successfully outperforms the CLC policy technique.	More QoS parameters like makespan and throughput also can be a part of the research.

2.3 Virtual Machine Generation in Fog System

In this paper, Samson Busuyi Akintoye et al. (Akintoye & Bagula, 2019) suggested a job allocation and virtual machine placement technique. In order to optimally utilize virtual machine and real machine resources, researchers concentrated on task allocation policy. Task assignment has been carried out using a binding strategy based on the Hungarian algorithm. In this study, the task assignment policy between cloudlet and VM is one-to-one. The Fog/Cloud environment has employed the GABVMP to enhance Quality of Service. The introduction of a self-developed GUI-based interface made it easier to define the parameters for cloudlets, virtual machines, and the method for allocating cloudlets to virtual machines during simulation. Using the CloudSim Simulator, researchers apply the suggested technique to measure latency, energy, processing time, and parameter allocation costs. A comparison was made between the simplex method and the suggested Hungarian algorithm for binding policy (HABBP) using default assignment strategy. HABBP has demonstrated a 54.73% reduction in overall processing time when compared to the default assignment technique. When comparing HABBP to a simplex method, calculation time has showed a good improvement. First-Fit placement and random placement were contrasted with the suggested method GABVMP. When it comes to parameter delay and the total cost of placing a virtual machine on a physical machine, GABVMP performs better than both First-Fit placement and Random placement. This work misses many tasks to a single machine allocation policy. When compared to the default assignment approach, HABBP does not demonstrate a faster computation time. Researchers intend to apply a suggested method in real-world traffic control or healthcare in Africa in the future.

The idea of enhancing QoS in a Fog environment by allocating resources effectively was presented by Sathish Kumar Mani et al. (Mani & Meenakshisundaram, 2020). In the Fog-Cloud system, researchers have worked on VM placement and workload distribution. A Fog Server Manager (FSM) is developed in the proposed system to allocate available resources to jobs. However, each Fog server has a single FSM and several VMs to handle user requests. Resources will be allocated if they are available at the end of the Fog server. If not, consumers will have to wait for state or deallocate requests to the cloud. When tasks are finished, the Fog server and cloud server will send an ACK to the FSM. Parameter computation time and processing time are simulated using the CloudSim tool. The suggested technique outperforms the default strategy on parameter processing time, according on the comparison between the default strategy and

the simplex algorithm. For some jobs, researchers are unable to outperform the default technique on parameter computation time.
In terms of range The busy-checking 2-way-balanced approach was developed by Sudip Roy et al. (Al-Tarawneh, 2022) as a method to allocate virtual machine resources to cloudlets in an efficient manner. Scholars have utilized three-phase methodologies to execute this approach: a cloudlet still not allocating phase, a VM classification phase, and a 2-round active checking phase. To ensure efficient allocation of virtual machines to cloudlets via a two-way approach, scholars have implemented equilibrium conditions for cloudlet distribution that strike a balance between the threshold and the local queue length limitation.In the CloudSim toolkit simulation configuration, scientists have conducted comparisons between conductance, Max-Min, Min-Min, RASA, and average VM utilization rate, as well as average makespan, average waiting time, average turnover time, and average VM allocation standard deviation. Researchers have surpassed all previously described methods on specific parameters; also, the algorithm's time complexity is O(mn) when compared to all previously mentioned algorithms' O(mn2). A comparison of this method with some of the most recent technologies is necessary, and researchers intend to use non-linear programming and soft computing in the future to create a suggested optimization method.

Table 2.3: Comparison Table of significant literature techniques for Virtual Machine Generation

Author name and Reference	**Type of Algorithm**	**Technique used**	**Evaluation tool**	**Performance Metric**	**Pros**	**Cons**
Samson Busuyi Akintoye (Akintoye & Bagula, 2019)	Meta Heuristic	Hungarian algorithm, Genetic algorithm-based Virtual machine Placement policy.	CloudSim	• Latency • Cost	Improvise Processing time and computation time.	Many tasks to one machine allocation policy are missed.

Table 2.3: Comparison Table of significant literature techniques for Virtual Machine Generation

Author name and Reference	**Type of Algorithm**	**Technique used**	**Evaluation tool**	**Performance Metric**	**Pros**	**Cons**
Sathish Kumar Mani (Mani & Meenakshisundaram, 2020)	Scheduling Algorithm	Acknowledgment-based Fog server manger	CloudSim	• Makespan	Enhanced computation time as compared.	Shortcomin in computation time.
Sudip Roy (Roy, Banerjee, Chowdhury, & Biswas, 2017)	Scheduling Algorithm	Range-wise Busy-checking 2 way-balanced	CloudSim	• Average waiting time • Average turnaround time • Makespan	Reduced Time complexity O(mn).	Compared with old technologies.

2.4 Smart Cities Application

The Min-Min conventional strategy to controlling Fog and cloud system resources in Smart Buildings was described by Saniah Rehman et al. (S. Rehman et al., 2018). The suggested method made use of six clusters, each comprising six areas with twenty buildings. Every region has a smart grid set up to forward each building's energy usage request to Fog. For each building, researchers have received 100 requests per hour for 128-byte data sets used in simulation setup. To effectively leverage the reservoir of the Cloud-Fog system, the simulation setup employed a service broker policy and a load management algorithm. For a simulation cloud analyst tool was used, and the authors' goal was to lower

the system's reaction time and cost. The cost was calculated using the data transport, micro grid, and virtual machine costs as inputs. A comparison of the suggested approach, and Round Robin algorithm was shepherd in order to evaluate the outcomes, and the latter proved to be less effective than the former. The suggested method works best for tasks that require the least amount of time to complete.

A method of job scheduling in Fog computing utilizing the Cuckoo optimization methodology has been reported by Saqib Nazir et al. (Nazir et al., 2018). Researchers have applied cuckoo optimization to the smart grid system; six areas, each with 20 buildings and 40–60 apartments per building, were taken into consideration. Each region is connected to two patches of Fog. Every apartment has created a request to contact the Fog nodes. The Fog node then verifies the availability of virtual machines (VMs) based on underutilized and overutilized nodes. A selection strategy will then be implemented for the VM allocation. Requests that are made when the VM is not available will be dispatched to the cloud server. The Cloud Analyst tool was utilized to build up the simulation, with average-processing-time,cost, and average-response-time parameters. The suggested method has been contrasted with Roun Robin and Throttled. On the specified parameters, the Cuckoo optimization technique (COA) has outperformed Round Robin and Throttled. Since the comparison is made between throttling and round robin, the simulation result does not indicate an improvement in the balanced average reaction time of six regions.

A plan for overseeing coupled resource management for Fog environment for smart cities was put up by Tian Wang et al. (T. Wang et al., 2019). When a single sensor receives several user service requests, a coupling resource management issue arises. To build up the suggested method, researchers expand the Hungarian algorithm with a buffer queue (EHGB). When coupling resource management, researchers try to reduce costs and delays while improving resource management. Users and resources are matched using the Hungarian algorithm. To increase resource consumption, however, resources that are still waiting will be allocated to users. Because buffer queues respond immediately to results, they are employed in Fog systems to prevent delays. One-half of the total resources are present in the buffer queue. Matlab 2016a and Visual Studio are utilize to assess the results. When the simulation outcomes were compared to those of the FIFO, Hungarian, and Extended Hungarian algorithms, it was found that the suggested strategy improved both the overall scheduling time and resource utilization. Left over resources are throwing the system into disarray due to an unbalanced distribution of users and resources.

Mutaz A. B. Al-Tarawneh(Al-Tarawneh, 2022) In order to position applications in the Fog environment, the researchers have devised a biobjective strategy. This method chooses where apps should be placed on Fog vertex based on the critical proportion and security mechanism. The NSGA-2 algorithmic is apply to resolve the bi-empirical knapsack formulation of the application placement issue.

Table 2.4: Comparison Table of significant literature techniques for Smart Cities Application

Author name and Reference	**Type of Algorithm**	**Technique used**	**Evaluation tool**	**Performance Metric**	**Pros**	**Cons**
Saniah Rehman (S. Rehman et al., 2018)	Heuristic	Min-Min	CloudAnalyst	• Response time • Cost	Improvise response time and cost.	Traditional Min-Min used.
Saqib Nazir (Nazir et al., 2018)	Meta Heuristic	Cuckoo optimization	CloudAnalyst	• Cost • Average processing time • Average response time.	Determined job scheduling efficiently.	Shortcoming in computation of response time.
Tian Wang (T. Wang et al., 2019)	Heuristic	Hungarian algorithm with buffer queue	Visual Studio and MATLAB	• Delay • Cost	Effective for handling multiple user requests by one sensor.	Resource provisioning mechanisms are not included.

Table 2.4: Comparison Table of significant literature techniques for Smart Cities Application

Author name and Reference	**Type of Algorithm**	**Technique used**	**Evaluation tool**	**Performance Metric**	**Pros**	**Cons**
Mutaz A. B. Al-Tarawneh (Al-Tarawneh, 2022)	Heuristic.	Bi-objective algorithm	iFogSim	• power efficiency • Application placement • Performance • Security rates	Improvement is noticed for all parameters.	Some highly scored QoS parameter was missing.

2.5 Load balancer in Fog computing

To reduce the TET, and TEC associated with dependent operations in a Fog environment, Sumit Bansal et al.(S. Bansal & Aggarwal, 2023) describe the PWOA method. Comprehensive simulations employing experimental instances (30, 50, 100, and 1000) through five distinct systematic effort—Cybershake, Epigenomics, Inspiral, Montage, and Sipht—are utilize to ingress the execution of the PWOA algorithm.Varying numbers of tasks were assigned to each of the workflows. The simulation conclusion showed that, in terms of raising TET and TEC throughout all processes, the suggested PWOA algorithm performed better than the traditional PSO and WOA algorithms.Additionally, this study presents a comparative assessment of the efficient workflow scheduling capabilities of the PWOA algorithm.

In their study, Bushra Jamil et al.(Jamil et al., 2020) provide a novel Fog computing scheduler that optimizes network use and minimizes delay, while also

facilitating service-provisioning for the Internet of Everything. To efficiently handle the demands of Internet of Everything devices on the resources of each Fog device, we present a case study that illustrates the optimal method for scheduling requests from these devices.Performance indicators such as delay and energy utilization are considered, and we utilize iFogSim to compare the proposed scheduling methodology with alternative alternatives. As demonstrated by the outcomes, the preferred scheduler reveal patronizing performance collated to the FCFS approach, achieving a 32% decrease in latency and a 16% enhancement in network utilization.

Samia Ijaz et al. propose a two-phase scheduling technique named Energy Makespan Multi-Objective Optimization, as described in their paper (Ijaz, Munir, Ahmad, Rafique, & Rana, 2021). The muddle is first constructed as a many-impartial optimization conundrum to start process (Abouaomar, Mlika, Filali, Cherkaoui, & Kobbane, 2021). INext, while assigning resources to Fog and Cloud, it computes a balance between competing goals.Essentially, it assigns latency-sensitive tasks that demand less processing power to computationally demanding procedures that must be coordinated with instantaneous response times from cloud resources. In order to attain an even greater energy savings, we complement the Deadline-Aware progressive Frequency Scaling strategy with empty time periods on a single node that occur between two previously scheduled jobs. Our solution effectively lessen energy utilization till 50% with no impact on schedule fulfillment as compared to current methodologies.Our research employing both synthetic and real-world applications forms the basis for this conclusion.

Pedram Memari et al. (Memari, Mohammadi, Jolai, & Tavakkoli-Moghaddam, 2022) aims to present a scheduling method that matches virtual machines using meta-heuristics and takes latency into account.Furthermore, the researchers want to integrate fog computing and the cloud to create an affordable energy management system for smart homes. Tabu search is often used in heuristic approaches because of its broad applicability to a broad scale of optimization trouble and its memory and speed advantages.Thus, a novel algorithm is suggested and enhanced through the utilization of FOA and approximate nearest neighbor (ANN) techniques. The Tabu search framework serves as the foundation for this algorithm. A final simulation of a case study is performed in order to validate the suggested methodology. The efficacy of the suggested method is then demonstrated through its development, consider critical elements for instance allocated memory, delay, execution time, and cost function. The comparison findings show that PSO, Tabu search, genetic algorithm, simulation-annealed, and all

other algorithms perform worse than the suggested approach.
The suggested algorithm is broken down into three phases in the research by R. Madhura et al. (Madhura, Elizabeth, & Uthariaraj, 2021). Task prioritization, task selection, and level sorting are the three separate phases of the procedure. An algorithm is suggested to rank jobs with the highest number of successors during task prioritizing, allowing excess jobs in the another step to start executing. The process of selecting tasks considers a combination of local and global optimal strategies to allocate a job to a preferred processor. This technique improves the choosing of processors by reducing the time it clasp to finish jobs and lowering the overall pattern it grabs to thoroughgoing all jobs. The comparison results clearly show that this algorithm works better than other used traditional approaches.By conducting multiple experiments involving both arbitrary-generated and real-life graphs, the efficacy of the provided algorithm is evaluated. As per the findings, the provided algorithm exhibits condescending accomplishment collated to all established algorithms, as sustained by performance matrices such as makespan, speedup, and average scheduling length ratio. This surpasses SDBBATS and minimal optimistic processing time.

2.6 Discussion

The relevant work that was discussed in this chapter are compiled in Table 2.1, 2.2,2.3,and 2.4. The parameters used in the methods and algorithms researchers have employed to solve resource management challenges, the kinds of tools or simulators employed in earlier studies, and the performance metric employed to assess the system have all been consider in the representation of the table. Following our analysis and summary of the relevant literature, we have come to the following conclusions: Many authors use the iFogSim simulator for configuring the Fog, and Cloud system along IoT devices, and many studies employed the meta-heuristic technique to resolve the scheduling of resources complication. Researchers most often utilize cost, makespan, latency, and throughput as metrics to assess system performance.

Chapter 3

Improved Grey Wolf Optimization Algorithm for cloud-Fog resource scheduling

Cloud platforms are where users and applications perform their computations in current span of cloud environment. The utility model of the cloud provides a variety of assistance, including IaaS, PaaS, and SaaS (M. Alam, Shahid, & Mustajab, 2024; Bhatt & Sehgal, 2024). These assistance are billed on a top up basis.Cloud-based information hub require a significant amount of electricity, leading to a rise in overall power usage. The initiation of the Fog computing model was proposed as a solution to address these difficulties (Aslanpour et al., 2020). The efficient distribution of computing resources in cloud-Fog systems, especially for time-sensitive applications, is a major challenge in the quickly developing digital landscape of today. The implementation of the "cloud-Fog paradigm" in distributed computing architecture relocates cloud functionalities towards the network's periphery, thereby enabling improved real-time application performance and reduced latency (Khan, Garg, Tiwari, & Upadhyay, 2018; Srirama, 2024). Depicted in Figure 3.1 is an example of a conventional fog computing architecture.

To direction the scheduling of resource complications in this setting, researchers created the "Improved Grey Wolf Optimization algorithm" for Scheduling of resources in a Cloud and Fog Environment for Delay-Sensitive Applications. This novel solution uses natural-inspired grey wolf behaviors to optimize the distribution of resources including computing power, network bandwidth, and storage

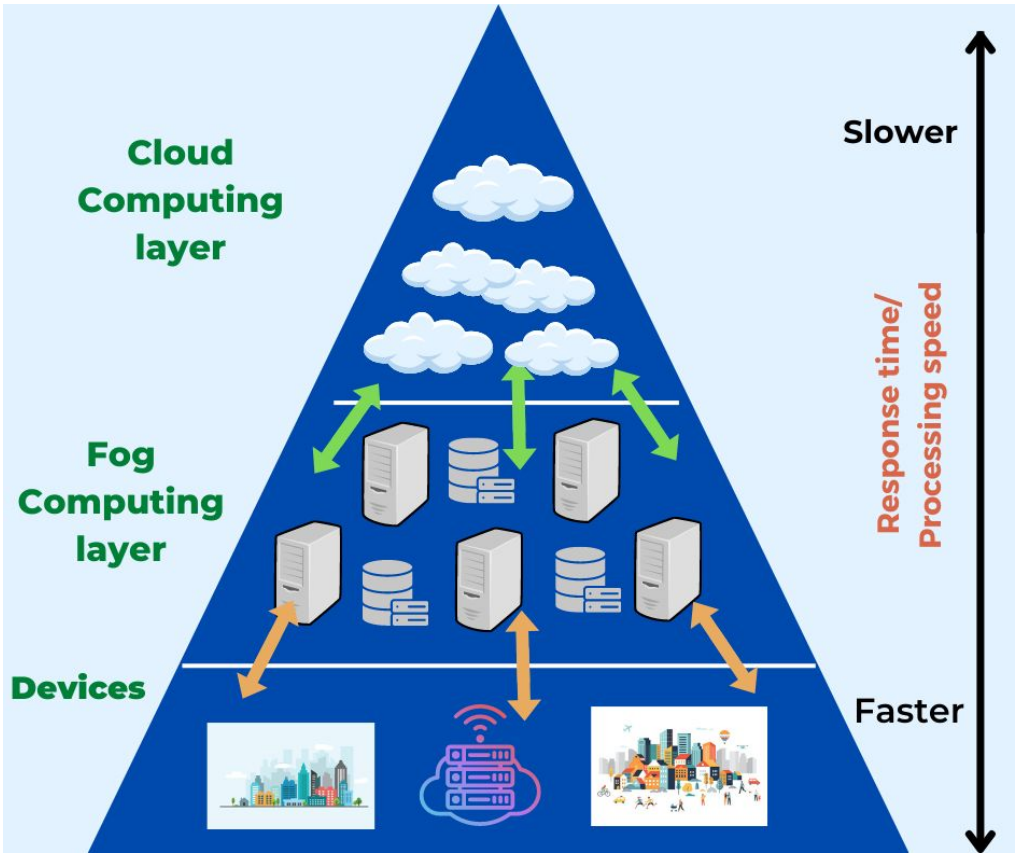


Figure 3.1: IoT-Fog-Cloud Environment

across a diverse and dynamic cloud-Fog ecosystem. This algorithm's major goal is to reduce response times and latency for delay-sensitive applications, delivering a seamless user experience and meeting demanding Quality-of-Service (QoS) criteria.

This work looks at the fundamental components and reasons behind the Improved Grey Wolf Optimization Algorithm, emphasizing its potential to transform resource scheduling and management in cloud-Fog scenarios. We will describe the difficulties associated with delay-sensitive applications, the basic concepts of the GWO algorithm, and how the enhancements increase its performance in the subject of cloud-Fog resource allotment. This method is a viable option for optimizing the resource allocation process, which will ultimately lead to increased efficiency and responsiveness in cloud-Fog environments, benefiting a wide range of requirements including real-life data analytics,IoT, and augmented reality. "Improved Grey Wolf Optimization Algorithm" is designed by improvising the Grey-wolf-optimization-technique (Alzaqebah et al., 2019) with a heterogeneous earliest finishing time (HEFT) (Dubey et al., 2018).In this work, details regarding obtainable resources of the Fog node is gathered by the fitness function. On each iteration, the Fog node fitness function will update. Then

the task can be issued to every node based on the compute EFT function. The IGWOA helps to find an effective solution in search space as per exploitation and exploration techniques. It avoids acquiring adhere in local optima, which helps in improvement of QoS requirements in the IoT-Fog-Cloud system (Kishor & Chakarbarty, 2021; Ramzanpoor et al., 2022a).

The key involvement of this work are as given below:

1. A novel optimization technique for resource allocation IGWO is designed and proposed for Fog environments.
2. The improved heuristic function is designed and used for the ideal allocation of jobs/tasks towards resources of Fog nodes and cloud servers.
3. Performance comparison is conducted using 3 standard datasets with peer competing for meta-heuristic optimization techniques like AEOSSA (Abd Elaziz et al., 2021), Harry Hawks Optimization (Heidari et al., 2019), PSO(Okwu, Tartibu, Okwu, & Tartibu, 2021b) and the FA (Yang, 2009) for QoS parameters including makespan and throughput.

Table 3.1 shows the notations used in our suggested solution .

3.1 Background and Detailed Description

This section concisely expresses the grey wolf optimization (GWO) and heterogeneous Earliest Finish Time (HEFT).

3.1.1 Grey Wolf Optimization Algorithm

The Grey Wolf Optimization Algorithm, which Seyedali Mirjalili (Okwu, Tartibu, Okwu, & Tartibu, 2021a) developed, is a metaheuristic algorithm that operates on population forms. It simulates the hunting and supervision mechanisms employed by wolves. Grey wolves live in groups and look for food in groups. Grey wolves follow a dominant hierarchy, as shown in Figure 3.2, for their groups of 5–12 members. Alpha is considered the head of the group; beta is the decision-maker candidate for the alpha; delta is the caretaker or dominates the omegas; and omegas come at last for eating. There are three key stages of GWO: the 1^{st} phase is looking for prey, the 2^{nd} stage is encircling the prey, and

Table 3.1: Notation and definitions

Notation	**Definition**
$\vec{X}_{\text{p}}(t)$	Prey-Position-vector
$\vec{X}(t)$	Current position of wolf
$\vec{A}$ and $\vec{C}$	Coefficient vector
$\vec{A}$	Distance vector
$rank_{\text{u}}$	Rank of the Task
$\widetilde{Wi}$	Task Execution Cost
$\widetilde{C}_{\text{i,j}}$	Communication Cost for the task
$ET_{\text{i,j}}$	Execution Time
η	Efficiency
U_{time}	Useful Time
T_{time}	Total Time
β	Bandwidth
Υ	Throughput
EST	Earliest-Start-Time
EFT	Earliest-finish-Time

the last phase is hunting the prey.

Description of an Algorithm

This section explains the fundamental components and dynamics of the grey wolf optimization Algorithm 1.

I. **Encircling the prey**

Prey will be encircled during hunting by grey wolves. Below mentioned mathematical equations are suggested for encircling the prey:

$$\vec{D}=\left|\vec{C}\cdot\vec{X}_p(t)-\vec{X}(t)\right| \tag{3.1}$$

$$\vec{X}(t+1)=\vec{X}_p(t)-\vec{A}\cdot\vec{D} \tag{3.2}$$

Algorithm 1: Grey wolf Algorithm pseudocode

Procedure Start:;

1. Create a wolf population with random positions.;
2. Determine the number of iterations to be performed ($Max_{\text{iterations}}$);
3. Define the search space range ($Search_{\text{space}}$);
4. **For** `iteration` within the range $(1, Max_{\text{iterations}})$;
   For each wolf in the population;
   Determine the fitness of the existing solution.;
   End For;
5. Arrange the wolves according to their fitness levels.;
6. Bring the alpha, beta, and delta wolves up-to-date ;
   α_{Wolf} = population[0];
   β_{Wolf} = population[1];
   δ_{Wolf} = population[2];
7. **For** each wolf in the population;
   Update wolf's position using the α, β, and δ positions;
   End For;
8. Implement Exploration and Exploitation;
   $a = 2 \cdot \texttt{rand()} \cdot \left(1 - \frac{\texttt{iteration}}{Max_{\text{iterations}}}\right) - 1$;
   $C1 = 2 \cdot \texttt{rand()}$;
   $C2 = 2 \cdot \texttt{rand()}$;
9. Compute distances for wolves α, β, and δ ;
   $\text{D}_{\alpha} = |2 \cdot C1 \cdot \texttt{Wolf-position}_{\alpha} - \texttt{wolf-position}|$;
   $\text{D}_{\beta} = \left|2 \cdot C2 \cdot \texttt{Wolf-position}_{\beta} - \texttt{wolf-position}\right|$;
   $\text{D}_{\delta} = |2 \cdot C2 \cdot \texttt{Wolf-position}_{\delta} - \texttt{wolf-position}|$;
10. Calculate the new position for the wolf;
    $\text{New}_{\text{position}} = \frac{\texttt{Wolf-position}_{\alpha} - a \cdot \text{D}_{\alpha} + \texttt{Wolf-position}_{\beta} - a \cdot \text{D}_{\beta} + \texttt{Wolf-position}_{\delta} - a \cdot \text{D}_{\delta}}{3}$;
11. Ensure the wolf's position is within the search space;
12. Update the wolf's position using $\text{New}_{\text{position}}$;
13. Assess the fitness of $\text{New}_{\text{position}}$;
14. The best solution found in the whole population as the optimal solution.;

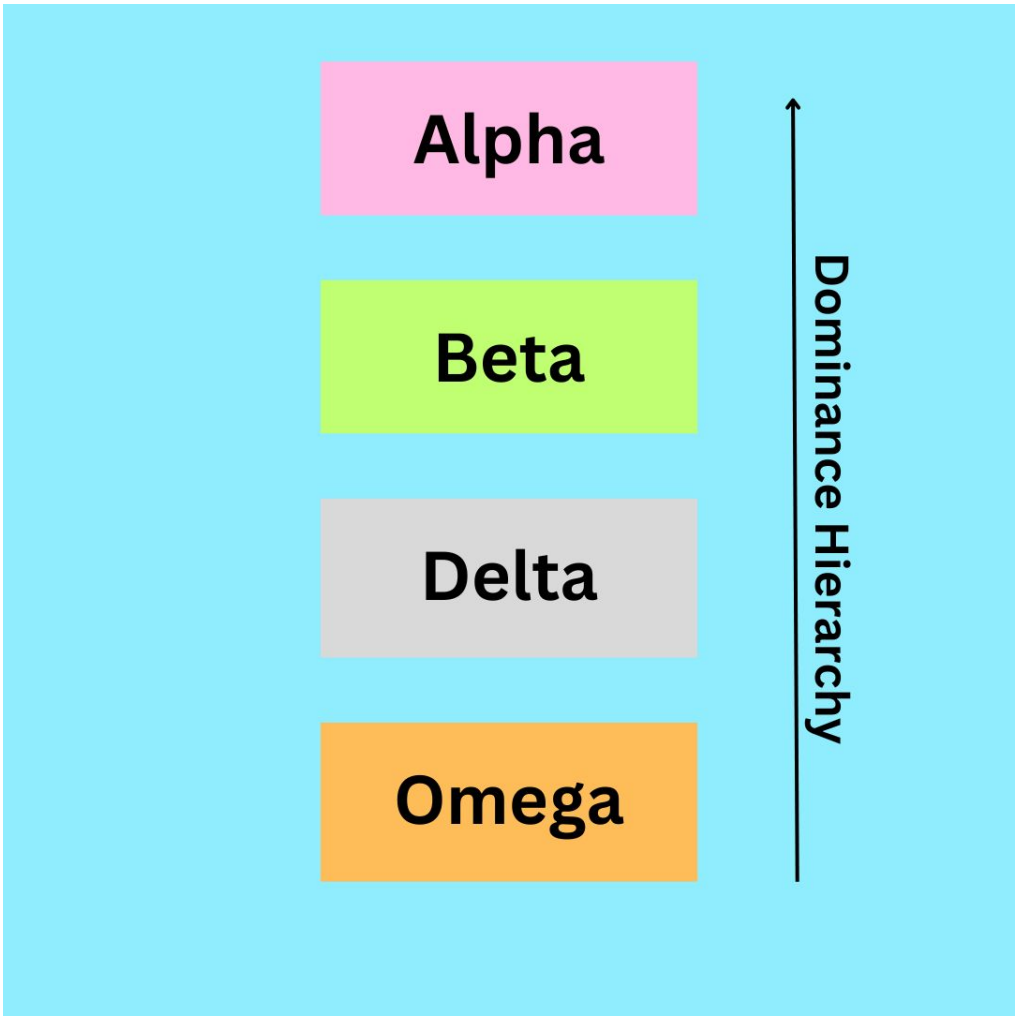


Figure 3.2: Grey Wolf algorithm Ranking mechanism

In the above equations 3.1 and 3.2, the current iteration is indicated by t, the coefficient vector is represented by $\vec{A}$ and $\vec{C}$, prey position vector is constituted by $\vec{X}_p$, and the grey wolf position vector is indicated by $\vec{X}$.

$\vec{A}$ and $\vec{C}$ vector is deliberated as follows:

$$\vec{A} = 2 \cdot \vec{a} \cdot \vec{r}_1 - \vec{a} \tag{3.3}$$

$$\vec{C} = 2 \cdot \vec{r}_2 \tag{3.4}$$

In above equation 3.3, $\vec{a}$ is lessen from 2 to 0 above the iterations, and $\vec{r}_1$& $\vec{r}_2$ vectors as shown in equations 3.3 and 3.4 lies between the [0,1].

II. **Hunting the prey**

Grey-wolves will encircle the prey and hunt them. All stalking of the wolf groups is escort by the alpha, and the beta, and the alpha have participated in hunting occasionally. As per the algorithm, we have considered $\alpha, \beta, and\delta$ to have the best knowledge of the prey, and ω wolves updated their location as per the location of $\alpha, \beta, and\delta$ wolves. The below-mentioned equations 3.5, 3.6, and 3.7 describe the discussed concepts:

$$\vec{D}_\alpha = \left|\vec{C}_1 \cdot \vec{X}_\alpha - \vec{X}\right|, \vec{D}_\beta = \left|\vec{C}_2 \cdot \vec{X}_\beta - \vec{X}\right|, \vec{D}_\delta = \left|\vec{C}_3 \cdot \vec{X}_\delta - \vec{X}\right| \quad (3.5)$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \cdot \vec{D}_\alpha, \vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \cdot \vec{D}_\beta, \vec{X}_3 = \vec{X}_\delta - \vec{A}_3 \cdot \vec{D}_\delta. \quad (3.6)$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \quad (3.7)$$

III. **Attacking the Prey (Exploitation)**

Grey-wolves will start hunting, instantly prey come to a stop moving. With the decrease in its value, wolves will approach prey for hunting. The value will vary in the range [-a, a]. When the value will vary in the range [-1, 1], then the contiguous stance of the search agent can be any stance between the current stance of the prey and the search agent.

IV. **Searching for the Prey (Exploration)**

As per the position of α, β, and δ, all other wolves improve their locations and search for prey. With |A|>1, all wolves separate from everyone for searching prey. Otherwise, with |A|<1, wolves are coming toward the prey. Value of |A| helps the wolves globally search the prey and avoid local optima during searching. Another factor $\vec{C}$, the value lies between the [0,2]. Value of $\vec{C}$ allows more levels of exploration. with the increase and decrease in the value of $\vec{C}$, avoidance of local optima can be seen in GWO.

3.1.2 Heterogeneous earliest finish time technique

HEFT is a heuristic-based scheduling technique for finishing tasks in the minimum amount of time on a heterogeneous system. HEFT is good for task execution, which yields a low makespan on the system (S. Gupta et al., 2022). HEFT works in 2 phases: the 1^{st} phase prioritizes the tasks, and the 2^{nd} phase

deals with choosing a suitable machine for tasks so that execution time is low. If two or more jobs/tasks have the identical precedence, then any random tasks among them will be assigned to machines. In order to implement the ranking mechanism, HEFT allocates a weight (w) to every node and edge through the calculation of average communication and computation costs, as described in equation below:

$$\text{ranku}(t_i) = \widetilde{W}_i + \max_{t_j \in \text{succ}(t_i)} \left(\widetilde{\zeta}_{i,j} + \text{ranku}(t_j) \right) \tag{3.8}$$

In above equation 3.8, ranku is the rank of task ti to the last node of the longest path. In the task(t_i, t_j), t_i is the entry node and t_j is the exit node. Between the path from t_i to t_j, all successors of ti will come.$\widetilde{W}_i$ is the implementation cost of task, succ(t_i) is the whole successor of task(t_i), $\widetilde{\zeta}_{i,j}$ is computed by the taking average of the transmission cost of the task. At last, the graph is traversed in an upward direction and $rank_u$ is allocated to every node in the path. The rank of exit node is equal to the average implementation cost of that node. After assigning the ranks to each node, tasks will be scheduled to machines in decreasing order of their rank.

Description of an Algorithm

The given Algorithm 2 describes the Heterogeneous Earliest Finishing Time (HEFT) task scheduling method for a parallel and distributed computing environment. By assigning tasks to available heterogeneous resources in the best possible way, this method search for to narrow the total volume of time that they take to complete. The following gives an explanation of each algorithmic steps.

I. **Data structure initialization:**

   Priority queue and empty schedule, which will be used to keep track of task assignments, are initialized at the beginning of the method.

II. **The Upward Ranks** (U_{Rank}) **for each task should be determined:**

   The upward rank (U_{Rank}) for each task in the $(Task_{Graph})$ is determined in this step.When considering the task's dependent tasks' execution periods as well as its own execution time, the(U_{Rank}) represents the longest possible time it will take for a task to complete. This knowledge aids in

Algorithm 2: HEFT Algorithm

Data: TaskGraph, AvailableResources
Result: Schedule
Initialization of Data Structures;
$priority_{Queue}$ = Create-$Priority_{Queue}$();
Schedule = Create.Empty-Schedule();
1. Determine the upward ranks (U_{rank}) for each assignment;
For every Task in $Task_{Graph}$ **do**;
U_{rank}[Task] = Calculate.U_{rank}(Task, $Available_{resources}$);
End For;
2. Determine the downward ranks (D_{rank}) for each assignment;
For every Task in TaskGraph **do**;
D_{rank}[Task] = Calculate.D_{rank}(Task, $Task_{Graph}$);
End For;
3. Determine the priority of each task;
For every Task in $Task_{Graph}$ **do**;
Priority[Task] = (U_{rank})[Task] + D_{rank}[Task];
End For;
4. Tasks should be prioritized;
For every Task in $Task_{Graph}$ **do**;
insert(Task, $Priority_{Queue}$, Priority[Task]);
End For;
5.**While** priorityQueue is not empty **do** ;
New_{Task} = remove.$HighestPriority_{Task}$($Priority_{Queue}$);
Best.Resource = Find.Best.Resource($Next_{Task}$, $Available_{resources}$, Schedule);
Schedule[$Next_{Task}$] = Best.Resource;
End While;
return Schedule;

work prioritization.

III. **Find the (D_{Ranks}) (Downward Ranks) for each task:**

This stage determines the task's downward rank (D_{Rank}), which is similar to(U_{Ranks}). (D_{Ranks}) represents the shortest time a task needs to finish, taking into account both its own execution time and the execution times of tasks on which it depends.

IV. **Calculate Each Task's Priority:**

Each task's priority is calculated by adding its (U_{Ranks}) and (D_{Ranks}). Higher priority tasks are more important and are probably scheduled earlier.

V. **Task Prioritization:**

Based on their computed priorities, all jobs are added to a priority queue. With the highest priority work at the front of the line, the priority queue makes sure that tasks are processed in order of priority.

VI. **Task Scheduling (While Loop):**

As long as the priority queue is not empty, the scheduling loop will continue. Each cycle includes:

(a) The task with the highest priority is taken out of the priority queue to choose $(Next_{Task})$.

(b) The best resource (processor) that is available for $(Next_{Task})$ is used to determine **Best.resource**. This stage entails taking into account the processing power of the available resources and maybe distributing the load among them.

(c) The specified resource is changed in **Schedule[NextTask]**, indicating that the task has been assigned to it.

VII. **Return Schedule:**

The algorithm then delivers the final schedule, which details which tasks are allotted to which resources and in what order, after scheduling all tasks.

3.2 Proposed Framework of IGWOA

In this segment, we are describing an autonomic approach to resource scheduling for a Fog-Cloud environment. Firstly, there is an allocation of tasks from the IoT device layer to the Fog domain (Duan et al., 2022). Then we provide an automatic task placement strategy as a solution to resource management problems. An improved grey-wolf technique is employed to deploy tasks to the Fog domain (Lyu et al., 2020).

A system model for deploying IoT device services to Fog Domain and the cloud server is described to fulfill the solution of the resource management problem in Figure 3.3. A framework is dissect into three layers: an IoT layer,Fog, and cloud layer. The IoT layer consists of an autonomous car, a tablet, and other small objects with sensing capabilities. These devices produce data and service requests, both of which must be quickly processed. The middle layer is in charge of effectively allocating work to cloud servers and Fog nodes. This layer includes a task scheduler, resource manager, and resource pool for handling the IoT device service request. The task scheduler is responsible for the allocation of tasks from available resources with the help of the resource manager. It guarantees that tasks are completed quickly and effectively. It retains the cumulative facts about the Fog nodes as per the defined scheduling policy, named Improved Grey-Wolf Technique. The cloud and Fog layer resources are monitored by the resource manager. It is essential in determining how to divide up jobs according to resource availability and other considerations. The resource pool is a collection of the cloud and Fog layers' accessible resources. It offers details on the resources that can be allocated to particular jobs to the task scheduler and resource manager. The system decides how jobs should be distributed among Fog nodes using a scheduling mechanism known as the "improved Grey-Wolf technique".

The improved algorithm helps find optimal solutions for resource management problems. As per the traditional grey wolf optimization approach, omega nodes

work as per the instructions of delta nodes. The proposed approach is modified by introducing a "heterogeneous earliest-finish time" in the Grey-wolf-optimization-technique. By using the HEFT technique on omega nodes, all omega nodes will get a chance in the exploitation phase, equivalent to delta nodes. The aim of the approach is to intensify resource utilization, reduce response times, and make sure that Fog nodes and cloud servers collaborate effectively to process data and services produced by the IoT. This improvisation helps the proposed approach find more solutions for resource management problems.

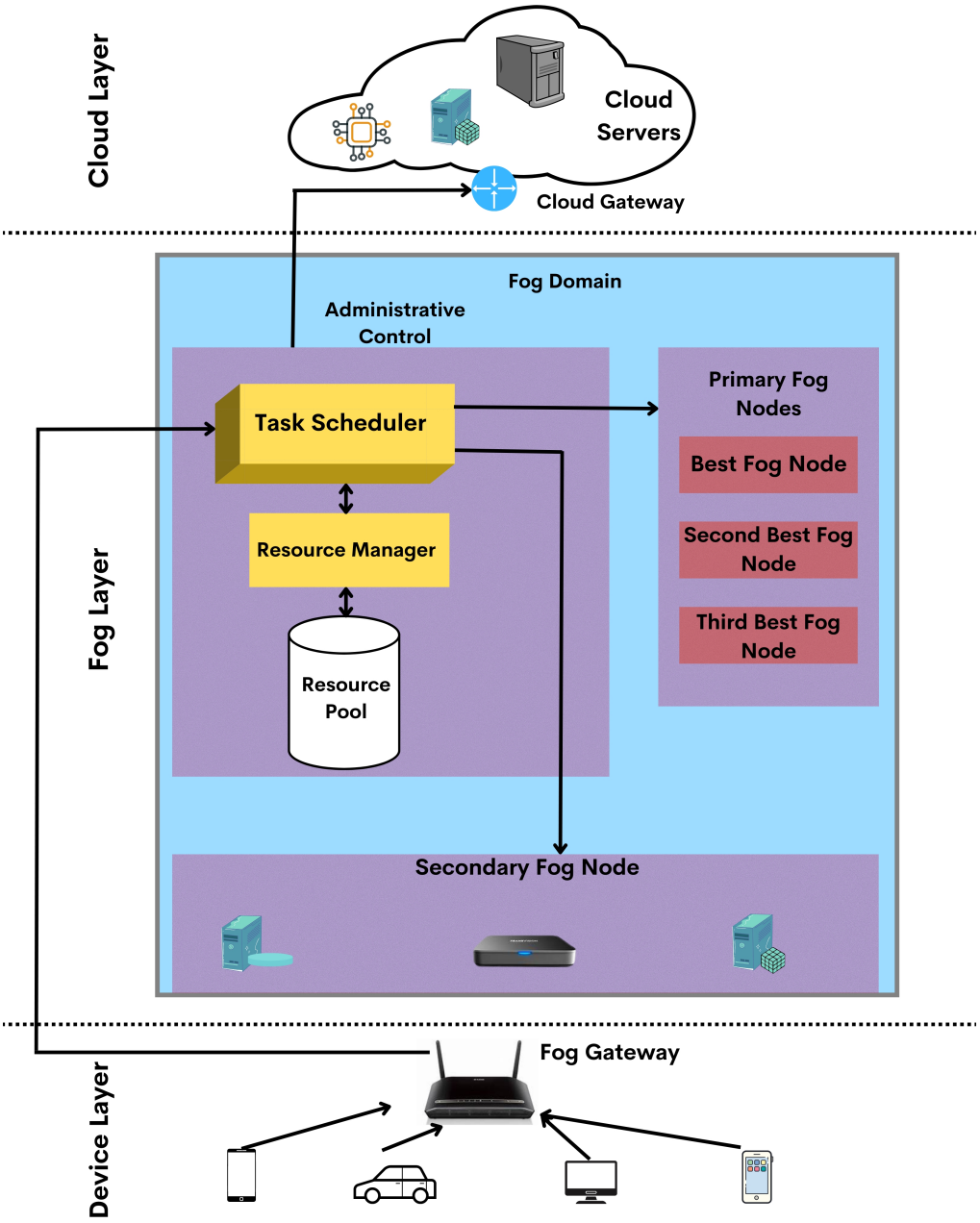


Figure 3.3: IGWOA Framework in Fog Environment

3.2.1 Detailed Description of Proposed Framework

This section explains the fundamental components and description of an architectural model.

I. **Fog layer** The Fog surface is the main surface in the proposed structure, located between the cloud surface and the IoT device surface. Fog layer includes three prime elements: administrative control, primary Fog nodes, and secondary Fog nodes, as exhibit in Figure 3.3. All three components described below are:

(a) **Administrative Control**

An administrative control system receives service requests from IoT devices from the Fog gateway. Requests are considered for as many tasks as possible by the system. Administrative control includes a task scheduler that schedules tasks to various nodes and servers as per scheduling policy while meeting QoS requirements. All resource information is available in the resource pool. The resource manager provides resource information to the task scheduler. As per available resources, the task scheduler decides whether tasks will be executed on either primary Fog nodes and secondary Fog nodes or cloud servers.

(b) **Primary Fog-nodes**

primary Fog-node domain consists of the three best Fog nodes: the best Fog node, the second-best Fog node, and the third-best Fog node. Primary Fog nodes are acting on the fundamental of the grey wolf optimization hunting mechanism. In the grey-wolf- optimization-technique α, β, and δ are considered best-fitting wolves from the candidate solution.

(c) **Secondary Fog-nodes**

Secondary Fog-nodes are nodes other than the primary Fog-nodes. These Fog-nodes are ω nodes except primary Fog nodes. As per the grey wolf technique, ω wolves are the least experienced individuals in the set. Similarly, secondary Fog nodes are the nodes with fewer resources as compared to primary Fog nodes.

II. **The Cloud Layer** The cloud layer comprises of a cloud gateway and servers at cloud. The cloud gateway forwards tasks that require higher computation and storage capabilities to cloud servers under administrative control. The cloud gateway also serves as a load balancer, distributing workloads among cloud servers in an equitable manner. In the cloud layer, cloud servers are outfitted with greater storage and processing capacities.

3.2.2 Proposed solution to problem and Implementation

This segment describes the mathematical establishment of the present approach to the problem of efficiently allocating of tasks to machines.Suppose Task (T_i)=T_1,T_2,T_3,... T_n) consists of n the quantity of activities that are required to execute on multiple machines in a Fog-IoT environment. Each task has its own number of parameters, like task length, file input and output, memory requirements, and deadlines. The size of tasks can be computed from the millions of instructions available in tasks. Fog environments, and cloud consist of n processors Task (P_i)= P_1,P_2,P_3,... P_n). Each processor, server, or node has its own set of heterogeneous resources like processor frequency, memory storage, and bandwidth. MIPS are used to measure the processing power of each CPU.
Matrices of the available number of tasks needed to execute on processors will be designed to compute the expected time of execution (ET) of the processor. Execution time will depend upon the length of task and the ability of computing nodes, as shown in equation 3.9. The ET of the task (Ti) is computed as mentioned below:

$$ET_{i,j} = \frac{\text{task}(T_i).\text{length}}{p_j.\text{power}} \tag{3.9}$$

As discussed earlier, tasks task (t_i).length can be computed by the number of instructions, and p_j. Power is the power of computing nodes in relation to MIPS (Millions-of-instructions-per-second).
In Fog environments, there is still problem of assigning tasks to machines efficiently. The focus is to detect the best solution to the problem so that there will be a minimization in completion time (makespan) and throughput. This work is focused on minimizing the makespan and throughput to resolve the Fog-cloud environment's task scheduling issue.
Makespan on the system as shown in equation 3.10 will be calculated as follows:

$$\text{Makespan}(X) = \max_{j\in\{1,2,\ldots,m\}} \sum_{k=1}^{n} ET_{i,j} \tag{3.10}$$

Throughput (Υ)-Throughput defines a system's efficiency on a given bandwidth. The efficiency of the system can be computed by equation 3.11, the equation describes the system efficiency by the useful time (U_{time}) on the Total time (T_{time}) for the system. Useful time (U_{time}) defines the timing for which the system performs without delay.

$$\eta = \frac{U_{\text{time}}}{T_{\text{time}}} \tag{3.11}$$

Throughput of the structure can be computed by equation 3.12, which defines the system efficiency over a given bandwidth (β).

$$\Upsilon = \eta \cdot \beta \tag{3.12}$$

In order to formulate the problem of task scheduling, one must minimize both the makespan and throughput. Conceptually, the solution to this issue can be delineated as follows:

$$\text{Objective-function}(F(Z)) = \min\left(\max_{j\in\{1,2,\ldots,m\}} \sum_{k=1}^{n} ET_{i,j}\right) + max(\text{Throughput}(\Upsilon)) \tag{3.13}$$

Main objective of this work as shown in equation 3.13, is to optimize throughput while minimizing makespan. For the purpose of optimizing the implementation of existing resources and inscription provocation associated to task scheduling.

3.2.3 The proposed improved Grey-wolf-optimization-algorithm

Proposed algorithm is created utilizing the HEFT approach, and Grey-wolf-optimization-algorithm. In the suggested algorithm, the exploration is enlarged by focusing on the grey-wolf-algorithm and exploitation is enhanced, because as per the proposed algorithm, all omega nodes do not depend on the delta nodes or third-best nodes for their hunting. In the proposed algorithm all omega or secondary nodes will get a chance equivalent to the delta nodes for hunting.

Algorithm 3: IGWOA Algorithm

Input: $Max_{iteration}$, list of brokers, list of machines, IoT tasks list.

Output: Optimal solution for Taskallocation to Brokers.

Start:;

1. Initialization:;
   (a) Initialize the grey wolf population X_i $(i = 1, 2, \ldots, n)$;
   (b) Initialize the parameters a, A, and C;
   (c) Compute the fitness of each search broker and categorize the broker as per Equation 3.13;
   (d) x_α = the best solution in the search space;
   (e) x_β = the second-best solution;
   (f) x_δ = the third-best solution;
   (g) Initialize $t = 0$;
2. **While** ($t <$ maximum number of iterations);
   (a) **For each search broker**;
      (b) Update the location of each broker by section I. of encircling prey;
   End For;
3. **If** (broker $\neq x_\alpha$ && broker $\neq x_\beta$ && broker $\neq x_\delta$);
   (a) Compute $rank_u$ for each task (t_i) by traversing the graph as in equation 3.8;
   (b) Sort the task (t_i) in the scheduling list in decreasing order of $rank_u$ values;

End If;

4. Updating the parameters a, A, and C;
5. Again, compute the fitness of all search brokers and categorize them;
6. Update the location of x_α, x_β, and x_δ and all other search brokers;
7. **While there is an unscheduled task** (t_i) on the list **do**;
   (a) Choose the first task (t_i) as per the highest $rank_u$ value from the list of scheduling;

End While;

8. **For each process** ($p_i = p_1, p_2, p_3, p_4, \ldots, p_n$) **in the processor do**;
   (a) Calculate $EST(T_i, P_i)$ and $EFT(T_i, P_i)$ values;
   (b) Assign Task (t_i) to processor P_i that minimizes EFT of task (t_i);

End For;

9. $t = t + 1$;

End While;

End;

Description of an Algorithm

This section explains the detailed description of proposed Algorithm 3 .

I. **Initial Stage**

In the above-mentioned algorithm, the initial phase is the initialization of the grey wolf population X_i (i=1, 2.... n) and parameters a, A, and C as mentioned in steps 1(a) and 1(b). A suitable initialization of the population is required for dealing with a issue with job scheduling in a cloud-Fog scenario. The modification in Grey Wolf is required to deal with problems of task scheduling. In step 1(c), the fitness of each machine will be calculated as per equation 3.13. Fitness calculation is required for generating ranks on the best-suited machine for computation in the system. In the above algorithm, steps 1(d)–1(f) provide the best machine with ranking α,β, and δ. In step 1(g), t is used to represent the current iteration of the iteration process. The iteration process will be stopped after the number of iterations reaches the maximum number of iterations as mentioned in step 2.

II. **Encircling the prey (updating stage)**

In the initial stage, the exploration phase is performed to search for prey and assign a rank to the best machines. The second phase of the algorithm is encircling the prey and initiating the best exploitation strategy, as shown in step 2(b). As per fitness calculation, those machines are not α,β, and δ. The HEFT algorithm is applied to them to improve the exploitation of the algorithm in steps 3(a)–3(b). On every iteration, parameters a, A & C, and the position of each broker will be updated to perform hunting as in steps 4-6 of an algorithm.

III. **Hunting the prey**

After encircling the prey, the third phase of the algorithm involves hunting the prey. In the final phase of the proposed algorithm, it assigns unscheduled tasks to the suited machines. In the traditional grey wolf algorithm, omega wolves are in control of delta nodes. Omega nodes are not getting an equivalent chance to hunt the prey. In the proposed algorithm, the HEFT mechanism is applied to nodes that are not α,β, and δ. So that they

are also getting a chance to hunting equivalents to other nodes. Steps 7-8 in the algorithm are working to allocate unscheduled tasks to the machines as per their ranking based on the HEFT algorithm. Step 9 is incrementing the iteration step by t=t+1.

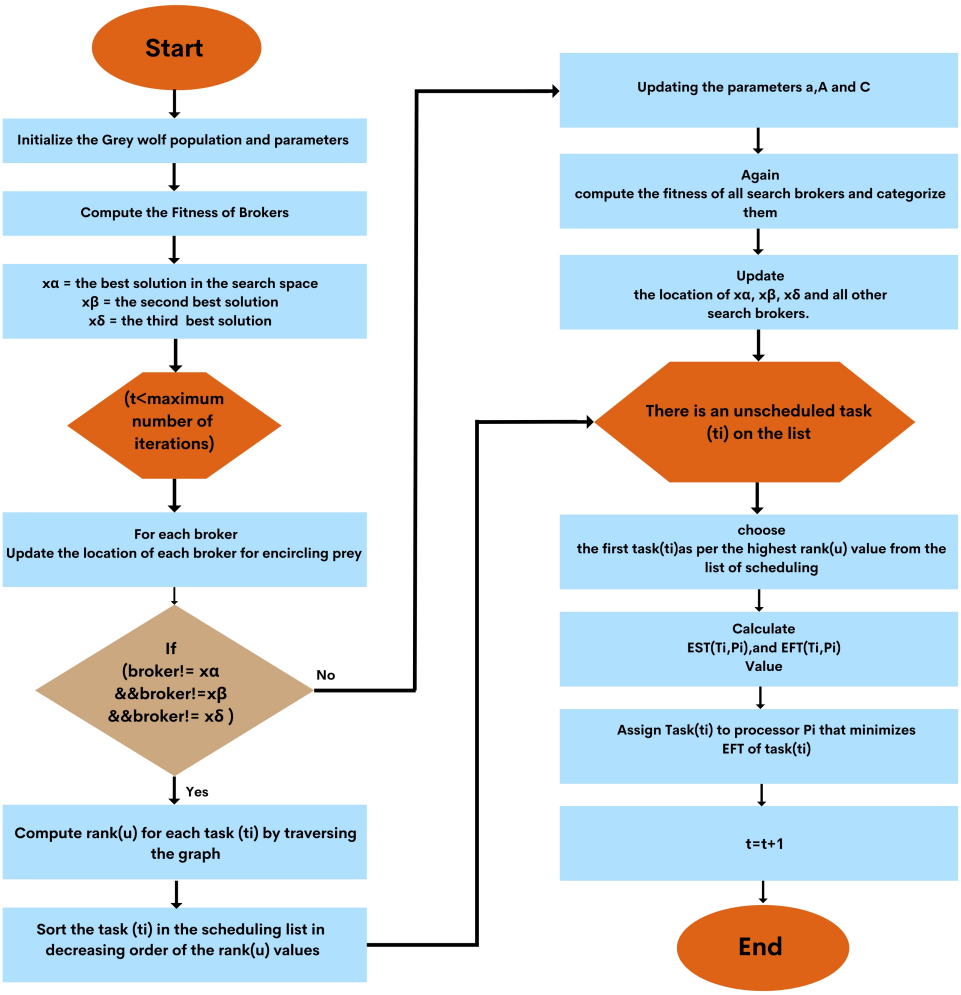


Figure 3.4: Methodology of improved Grey Wolf Algorithm

The proposed improved grey wolf algorithm is represented in Figure 3.4. Improvisation is done in the exploitation phase of the algorithm, where an unscheduled task is efficiently allocated to available resources in the cloud-Fog system. A designed objective function is shown in equation 3.13, which is calculated on the makespan and throughput by using the grey wolf and HEFT algorithms of the system. Related steps get repeated until the best solution is obtained and stop conditions are reached.

3.3 Experimental Evaluation

This segment, gives a information of the experimental setup, tests, and evaluation of results is done. The suggested approach IGWOA is evaluated and compared with existing approaches AEOSSA (Abd Elaziz et al., 2021), HHO (Heidari et al., 2019), PSO(Okwu et al., 2021b), and FA (Yang, 2009) for measuring the efficacy of the suggested approach.

3.3.1 Exploratory Environment and Experimental parameters

This segment describes the environment or parameters set during the experimental evaluation. The comparison and efficacy of the suggested approach are evaluated using the iFogSim toolkit (H. Gupta, Vahid Dastjerdi, Ghosh, & Buyya, 2017). All the examination are conducted on a constant and real dataset with a Core i5-2.40 GHz processor and 8GB of memory on a system equipped with Windows 10 64-bit.

Table 3.2: Cloud-Fog Entities

Entities	**Value**
Clients	[50,100]
Cloud server	2
Broker	4
Broker Storage capacity	1TB
Broker Bandwidth	10Gb/s
Broker RAM Size	16GB

Table 3.3: Task Entities

Entities	**Value**
Number of tasks	[200,1000]
Tasks Length	[1000,20000] MI
File Size	[300,600] MB
Output Size	[300,600] MB

Specifications of the constant environment for client, cloud, broker, and tasks are shown in Tables 3.2 and 3.3. For experimental evaluation, the number of clouds and Fog nodes is set to 2 and 4. Broker storage capacity, broker bandwidth, and broker RAM size are set to 1TB, 10 GB/s, and 16 GB. All experiments are done for 200 to 1000 tasks, with each task length used [1000, 20000] MI. To get accurate results, each experiment is conducted for 30 iterations.

Table 3.4: Description of the real dataset

Dataset Name	**Tasks**	**Users**	**CPUs**	**File Size**
NASA iPSC	18239	69	128	204kb
HPC2N	202871	257	240	2.9MB

For the experimental setup, pseudo datasets, NASA iPSC, and HPC2N (production system, july 2020) are used to show the efficiency of the proposed IGWOA. These datasets are provided by parallel workload archives. The log files are presented in original and cleaned versions by the community. All workload files are available with the extension.swf (standard workload format). In this work, a cleaned version is used to evaluate the approach, as shown in Table 3.4.
Makespan and throughput are evaluation metrics used in this work, as discussed in Section 3.2.2. An objective function described in equation 3.13 is used to compute the efficiency of machines. Machines with improved makespan and throughput values are assigned to scheduled tasks as per the proposed approach.

3.3.2 Results examination

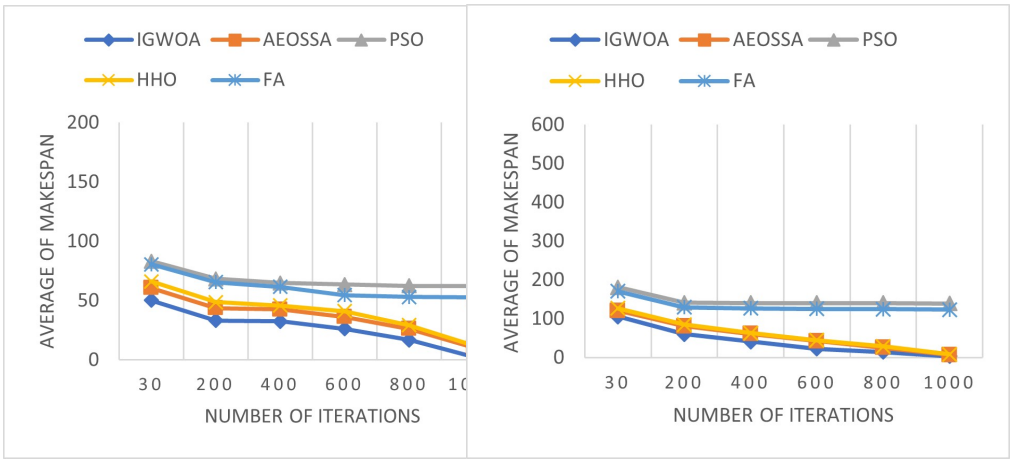
In this subsegment, experimental results processed by the proposed approach IGWOA are analyzed, tested, and compared with AEOSSA, HHO (Harris-Hawks-optimization), PSO (particle-swarm-optimization), and FA (Firefly algorithm). Results are evaluated on performance metric parameters like makepan and throughput based on the defined objective function. Results are evaluated and compared with the techniques mentioned for finding optimal solutions from global space. Table 3.5 presents the variable environments of the proposed and compared approaches.

Table 3.5: variable environments of proposed and compared approaches.

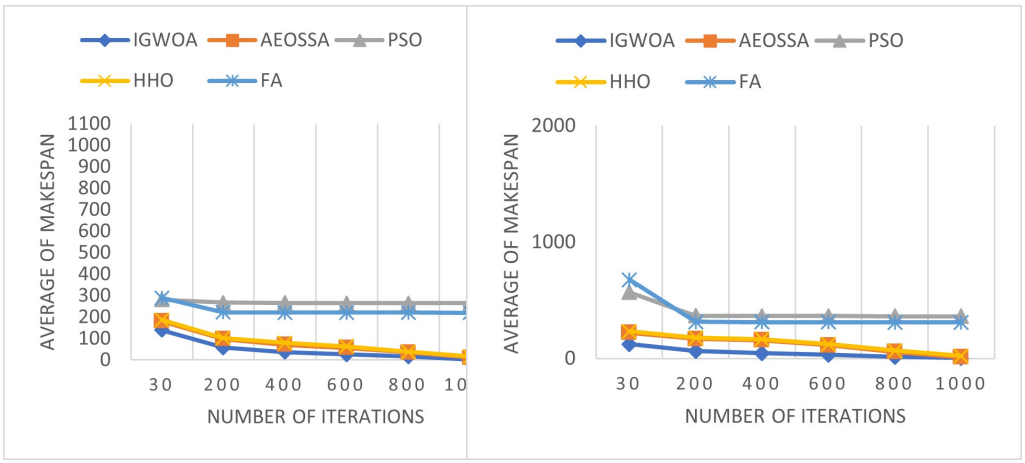
Algorithm	**Variables**	**Value**
IGWOA	$\mathbf{a}$	$[0, 2]$
	$\vec{r}_1$ and $\vec{r}_2$	$[0, 1]$
	$\mathbf{A}$	$[-1, 1]$
	$\mathbf{C}$	$[0, 2]$
AEOSSA	Swarm size	100
	rand1, rand2,rand3,and rand4	$[0, 1]$
	c1, c2, and c3	$[0, 1]$
HHO	Swarm size	100
	$E0$	$[-1, 1]$
PSO	Swarm size	100
	Inertia weight w	$0.9 \rightarrow 0.4$
	$c1$	1.49
	$c2$	1.49
FA	Swarm size	100
	γ	1
	β	0.2
	α	0.5

For checking the convergence manner of the proposed algorithm IGWOA, the convergence curve of the proposed algorithm is compared with AEOSSA, HHO, PSO, and FA in Figures 3.5, 3.6, and 3.7. The curve represents the frequency and quickness with which the algorithms deliver the best results. In comparison, IGWOA successfully outperforms all traditional techniques. In all comparisons, IGWOA represents the highest convergence rate and successfully provides optimal solutions from search space in comparison to other traditional techniques.

In Figure 3.5, a comparison is done with the number of iterations and average makespan on 200–1000 tasks for a pseudo-workload. As comparison has shown in all cases mentioned, on average makespan, IGWOA successfully outperforms all approaches. An improvement of 7.51%-9.34% is noticed over the AEOSSA, 4.69%-5.67% over the HHO, 68.73%-72.56% over the PSO, and 56.87%-65.67% over the FA on the pseudo dataset. The proposed strategy out-



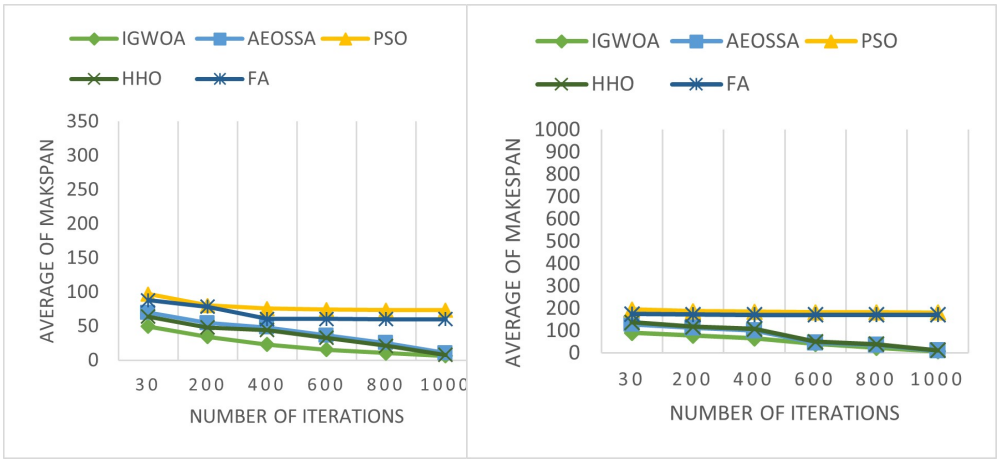
(a) 200 Tasks (b) 400 tasks



(c) 600 Tasks (d) 800 Tasks

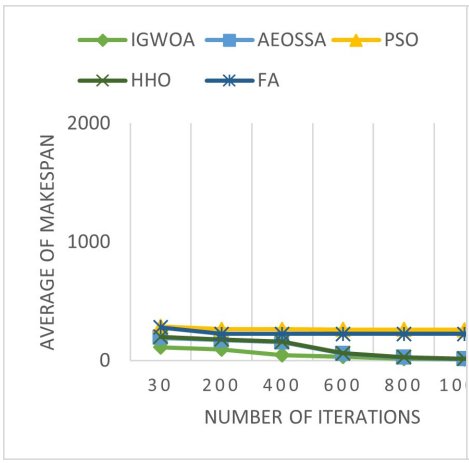
(e) 1000 Tasks

Figure 3.5: Curves of convergence for pseudo workload

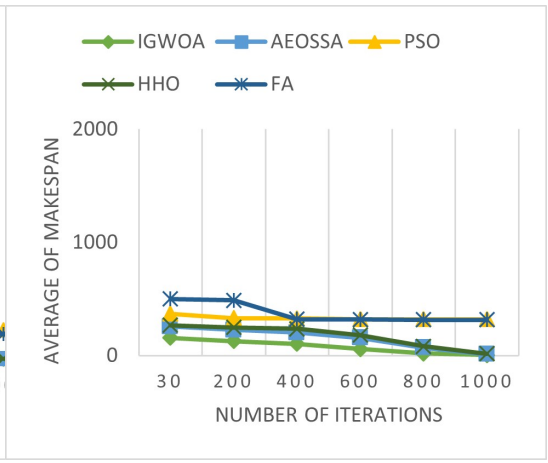


(a) 500 Tasks

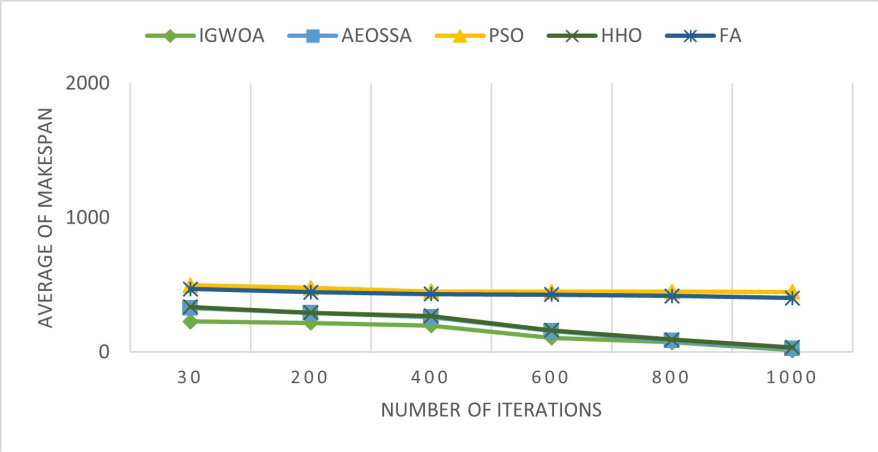
(b) 1000 tasks



(c) 1500 Tasks

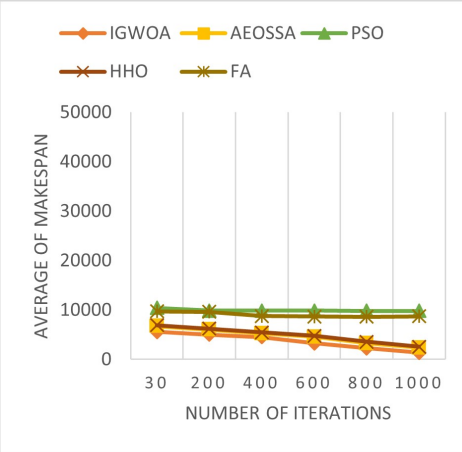


(d) 2000 Tasks



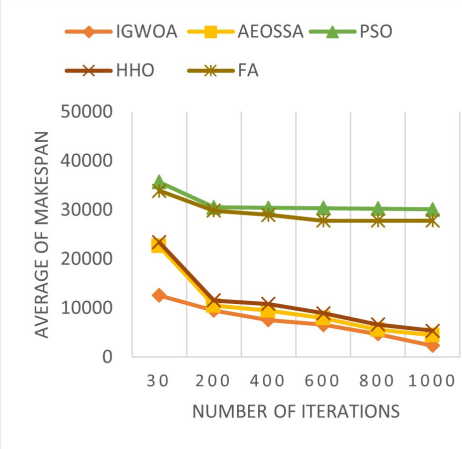
(e) 2500 Tasks

Figure 3.6: Curves of convergence for real workload NASA iPSC

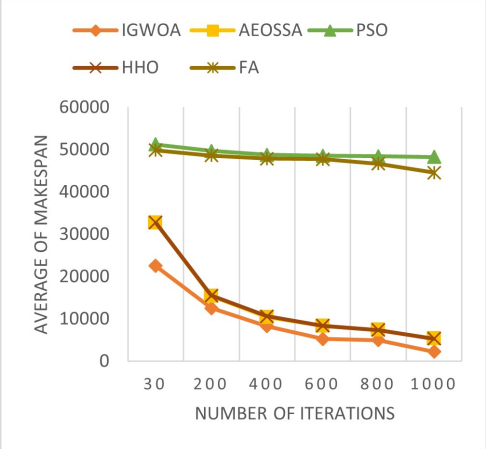


(a) 500 Tasks

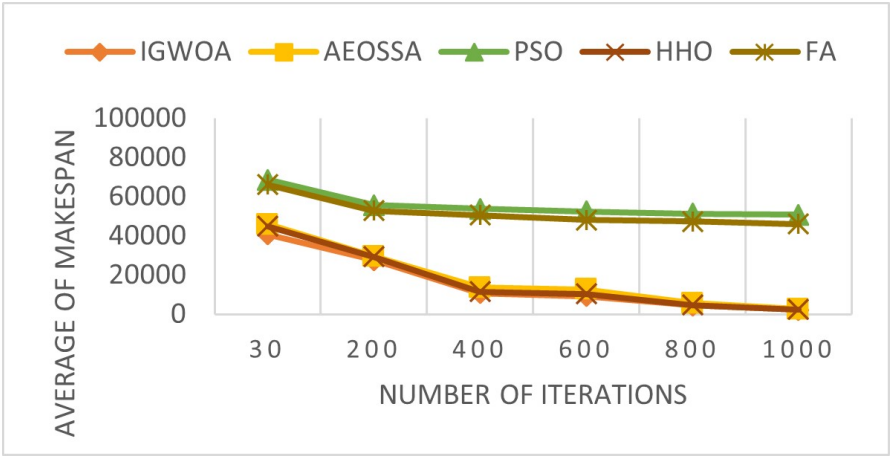
(b) 1000 tasks



(c) 1500 Tasks



(d) 2000 Tasks



(e) 2500 Tasks

Figure 3.7: Curves of convergence for real workload HPC2N.

performed conventional methods for tasks with a size of 200 to 800 with better improvement, while the convergence curve improved less for tasks with a size of 1000. Figure 3.5 shows evaluation during 30–1000 iterations. When the suggested technique runs for 1000 iterations, the makespan value decreases for various job sizes. there is a potential that resources will be less available in subsequent iteration steps. IGWOA's superior optimization tactics and search space exploration methods—which are more effective than those used by other techniques—allow it to produce better outcomes with a greater convergence rate. The enhancement over traditional techniques indicates that IGWOA's search techniques are better able to converge on optimal solutions. The superiority of IGWOA over AEOSSA, HHO, PSO, and FA on pseudo datasets implies that it can better balance exploration and exploitation, resulting in better overall optimization outcomes.

In Figure 3.6, a comparison is done with the number of iterations and average makespan on 500–2500 tasks for a real NASA iPSC workload. When a comparison is made with the NASA iPSC real dataset, an improvement of 5.34%–6.73% is noticed over the AEOSSA, 2.28%–9.67% over the HHO, 63.83%–69.73% over the PSO, and 47.23%–58.69% over the FA. In Figure 3.7, a comparison is done with the number of iterations and average makespan on 500–2500 tasks for a real HPC2N workload. Similarly, on the HPC2N dataset, an improvement of 4.42%–6.89% is shown over the AEOSSA, 64.96%–68.56% is shown over the PSO, 54.59%–59.76% over the FA, and 3.47%–4.68% over the HHO.In contrast to HHO, AEOSSA, and proposed approaches, FA and PSO optimization strategies have a greater value for 30-1000 iterations across 500–2500 jobs, as shown in Figures 3.6 and 3.7. When compared to PSO and FA approaches, HHO and AEOSSA techniques perform better in an exploratory environment as set in the system as established for outcomes evaluation.on HPC2N real dataset for customized environment, a deviation is noticed for AEOSSA, and HHO technique from 1000 tasks-2500 tasks over 200 iterations then stability is noticed after 200 iteration -1000 iterations.IGWOA has proven to be effective in addressing real-world data and optimization difficulties by producing better results with a higher convergence rate. The variation in improvement percentages raises the possibility that the performance of IGWOA may be influenced by the particulars of the problem instances within the real dataset. This significant improvement demonstrates IGWOA's superiority over traditional techniques on the NASA iPSC and HPC2N Real Dataset in terms of convergence rate and solution quality.

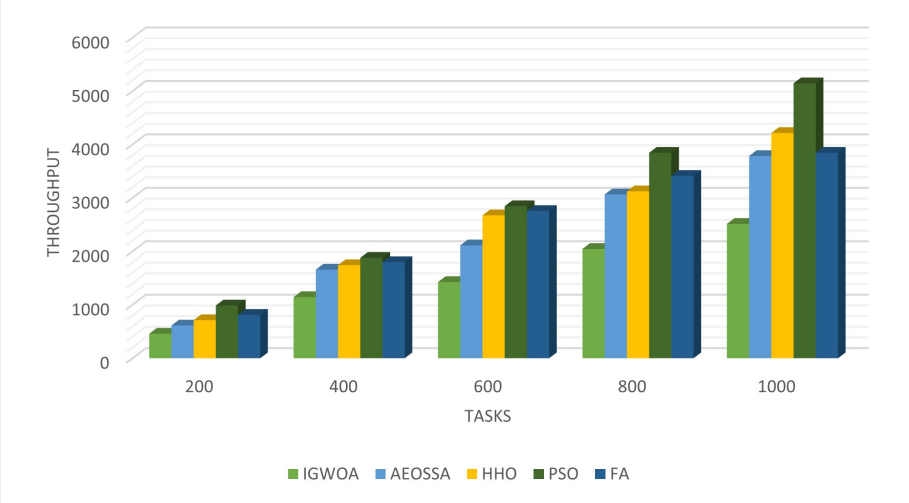


Figure 3.8: Average throughput time of pseudo Workload

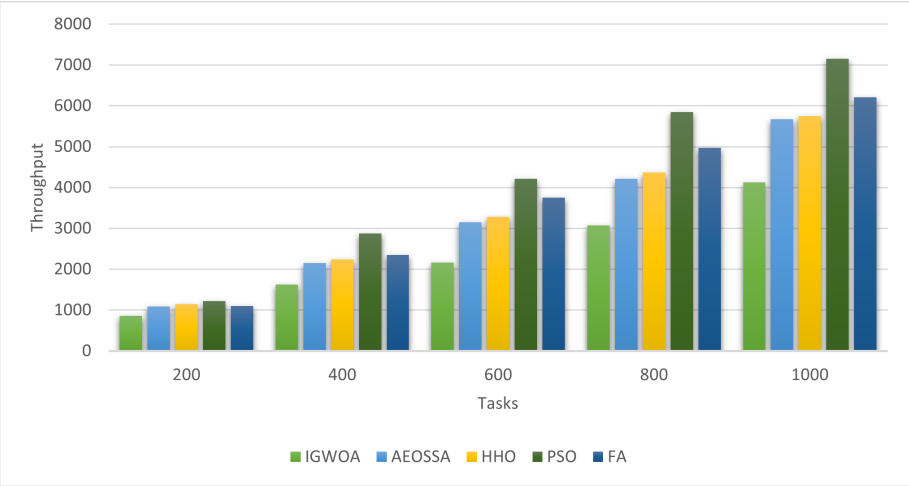


Figure 3.9: Average throughput time of NASA iPSC real workload

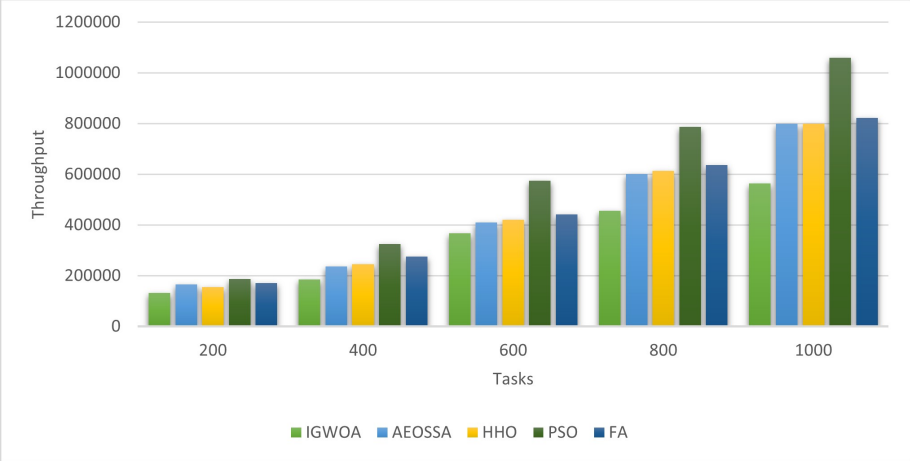


Figure 3.10: Average throughput time of HPC2N real workload.

Figures 3.8,3.9,3.10 represent the comparison of average throughput time for all datasets on 200–1000 tasks. The comparison is done on the number of tasks and throughput. The proposed approach achieved a better average throughput time in comparison to AEOSSA, HHO, PSO, and FA algorithms. The achieved results have shown the effectiveness of the proposed approach compared to traditional approaches. In Figure 3.8, a comparison is done on a pseudo-dataset on parameter throughput, and an improvement of 57.6%–62.4% is noticed among them. Figure 3.9 represents a comparison of average throughput time on NASA iPSC datasets; an improvement of 49.3%–52.8% is shown over traditional algorithms among the proposed approaches. Similarly, Figure 3.10 shows a comparison of the HPC2N dataset, and among this, a 35.2%–41.6% improvement is seen over the traditional technique among the proposed approach. IGWOA effectively explores the solution space by using a search method. By properly exploring and utilizing the search space, this tactic can assist the algorithm in finding better solutions. The outcomes repeatedly show that, for average throughput time, the recommended IGWOA technique outperforms the traditional techniques. These increases in average throughput time show IGWOA's ability to improve system performance, speed up task completion, and help with Fog computing activities. In Summary, the in-depth analyses and data repeatedly show that IGWOA is very successful in maximizing IoT task scheduling in Fog computing environments. On a variety of datasets, including synthetic and real-world data, it regularly outperforms conventional optimization algorithms. This suggests that IGWOA can schedule IoT jobs more effectively, resulting in enhanced system functionality and quicker task completion.

3.4 Statistical validation for IGWOA

ANOVA statistical analysis for three datasets on different task lengths has been done in Table 3.6. An analysis of all three datasets, with 1000 tasks for the pseudo dataset, 2500 tasks for the NASA iPSC dataset, and 2000 tasks for the HPC2N dataset, is performed. In all three analyses, the efficiency of the proposed algorithms is noticed over traditional algorithms. In statistical analysis, a p-value is less than 0.05 and the value of F crit is less than F, which shows the effectiveness of the suggested strategy. According to this analysis, the proposed approach has proven better for tackling optimization problems on the cloud-Fog system.

Table 3.6: ANOVA Statistical Analysis for three datasets on different tasks length.

Source of Variation	**SS**	**df**	**MS**	**F**	**P-value**	**F crit**
Rows	**158482.2**	**5**	**31696.44**	**9.883326**	**7.09E-05**	**2.71089**
Columns	**185642.1**	**4**	**46410.52**	**14.47135**	**1.05E-05**	**2.866081**
Error	**64141.24**	**20**	**3207.062**			
Total	408265.5	29				

(a) ANOVA Statistical Analysis for the pseudo workload on 1000 tasks

Source of Variation	**SS**	**df**	**MS**	**F**	**P-value**	**F crit**
Rows	**134754.7**	**5**	**26950.95**	**9.832376**	**7.34E-05**	**2.71089**
Columns	**545991.8**	**4**	**136497.9**	**49.79784**	**4.04E-10**	**2.866081**
Error	**54820.83**	**20**	**2741.041**			
Total	735567.3	29				

(b) ANOVA Statistical Analysis for real NASA iPSC dataset on 2500 tasks

Source of Variation	**SS**	**df**	**MS**	**F**	**P-value**	**F crit**
Rows	**9.12E+08**	**5**	**1.82E+08**	**8.926269**	**0.000139**	**2.71089**
Columns	**9.61E+09**	**4**	**2.4E+09**	**117.5306**	**1.36E-13**	**2.866081**
Error	**4.09E+08**	**20**	**20441738**			
Total	1.09E+10	29				

(c) ANOVA Statistical Analysis for real HPC2N dataset on 2000 tasks

Chapter 4

Whale Earthworm Optimization Algorithm for Load Balancing in IoT-Fog-Cloud Environment

In Industry 5.0 and the next generations, the IoT has a remarkable impression on network and computing technologies. The Internet of Things and its connect-ed components, such as sensors and network connectivity to various gadgets and house-hold things, enable access to many applications such as healthcare, traffic control, and self-driving automobiles, among others (Ramzanpoor et al., 2022a; Ramzanpoor, Hosseini Shirvani, & Golsorkhtabaramiri, 2022b). A variety of information are being generated by smart machines that must be prepared and computed for clients and end users in the least time to remain relevant with the speed of technologies.

In support of the evolution of the IoT and as an adjunct to cloud,and Fog computing is introduced. In order to mitigate the issue of latency (Gorlatova, Inaltekin, & Chiang, 2020; Tiwari, Sille, Salankar, & Singh, 2022), the Fog nodes extend the cloud layer to the periphery of the system, in closer proximity to the locations where IoT data is generated. To overcome various problems like latency, energy efficiency, and security data are prepared at Fog-nodes and data are moved to the cloud for high repository and more computational requirements. Evaluating paradigms Cloud and Fog are required for all kinds of IoT data. For enhanced QoS requirements, the Fog-Cloud paradigm is one of the best solutions available at the time(Mehmood et al., 2019; Qureshi et al., 2020).

A Fog system always deals with the constraint of processing capacity, unlike a cloud system. Many IoT applications are latency-sensitive and have latency-

tolerance with different types of requirements. Due to latency problems, scheduling, and management of these types of applications get difficult and not accepted in fields of healthcare and autonomous driving or real-time data processing requirements in the gaming industry. Resource scheduling is a solution to dealing with these types of problems(Aslanpour et al., 2020; Abouaomar, Cherkaoui, Kobbane, & Dambri, 2019). Communication and computation cost also becomes higher for various IoT applications due to the usage of cloud platforms. Computation time and storage are two factors that may affect the pricing of using a Fog-Cloud environment. Many existing techniques are unable to tackle the balance between the cost and Quality of Service (QoS) parameters.
Therefore, in this work, we are considering the task's significance with QoS and cost when they are scheduled in a Fog-Cloud environment. For scheduling tasks efficiently, a hybrid optimization algorithm based on the Whale-optimization-algorithm(Albert & Nanjappan, 2021) and earthworm-optimization-algorithm (Kumar & Karri, 2023) named Whale Earthworm Optimization Algorithm(WEOA) is proposed. The proposed algorithm is designed to achieve job scheduling which has cost benefits and efficient QoS attainment.
The main contribution of this work is as follows:

1. A hybrid Whale optimization and Earthworm optimization algorithm is proposed named WEOA to implement resource management efficiently in the Fog-Cloud environment.

2. The exploration phase and convergence speed are improved for optimal resource allocations.

3. Proposing and designing autonomous "Load-balancer based task Allocation Frame-work" as per the three-layer architecture of Fog environment.

4. Validating proposed work with many experiments on performance metrics like cost, makespan, and response-time.

As shown in Table 4.1, we provide the notations utilized in our recommended solution.

Table 4.1: Notation and definitions

Notation	**Definition**
$\vec{W}$	Position vector of whale
$\vec{W}_p$	Current position of Whale
$\vec{A}\&\vec{C}$	Coefficient vector
$\vec{P}$	Position of the Prey
F_{time}^{ti}	Finishing time of the Task
S_{time}^{ti}	Starting time of the Task
t	Current Iteration
$\vec{D}$	Distance
$T_{\max}$	Maximum iteration
NP	Earthworm Population Size
n_{KEW}	Number of earthworms
ψ_{cost}	Total Cost
$\psi_{\mathrm{comp-cost}}$	Computation Cost
$\psi_{\mathrm{comm-cost}}$	Communication Cost
$\psi_{\mathrm{cost}}(m)$	Cost of Memory
$\psi_{\mathrm{cost}}(p)$	Cost of CPU
$\psi_{\mathrm{cost}}(b)$	Cost of Bandwidth
MS	Makespan
RT	Response Time
Wt_{time}	Waiting Time
ART	Average Response Time

4.1 Background information and Explanation

The whale-optimization-algorithm (WOA) & Earthworm optimization method and autonomic computing procedure are briefly explained in this section.

4.1.1 whale optimization algorithm

WOA is a meta-heuristic optimization algorithm 4 that draws inspiration from biology. The WOA is established on the humpback whales', and bubble net attack procedure. One of the three methods—random search, shrinking encircling, or spiral feeding—is used to update the locations of search agents.
The humpback whales can track the surface of a diminishing circle & spiral path to detect the location of the prey while swimming around it. Thus, there are two methods of exploitation. The following is the equalization 4.1 and 4.2 for Shrinking Encircling:

$$\vec{W}(t+1) = \vec{W}(t) - \vec{A} \cdot \vec{D} \tag{4.1}$$

$$\begin{aligned} \vec{D} &= |\vec{C} \cdot (\vec{W}_p)(t) - \vec{W}(t)| \\ \vec{A} &= 2\vec{a} \cdot \vec{r} - \vec{W}(t) \\ \vec{C} &= 2\vec{r} \end{aligned} \tag{4.2}$$

Over the iterations, vector a $\vec{a}$ decrease from 2 to 0, while r vector $\vec{r}$ reclines between [0,1].
Between the present position and the prey, a spiral updating position is followed. The formula is as follows in equation 4.3:

$$\vec{W}(t+1) = \vec{D} \cdot e^{bL} \cdot \cos(2\pi L) + \vec{W}^* \cdot t \tag{4.3}$$

where b guarantees a logarithmic shape and l is a random value within the range [-1, 1]. The Random search method of the whale, which is chosen by A, ensures algorithm exploration. Thus, the equation 4.4 for the investigation is as follows:

$$\vec{W}(t+1) = \vec{W}_{\text{rand}}(t) - \vec{A} \cdot \vec{D} \tag{4.4}$$

Algorithm 4: Algorithm: Whale Optimization Algorithm

Input: Tasks and resources R for a Fog node F_{nod} where $F_{\text{nod}} = \sum_{c=1}^{c'} F_{\text{nod}c}$

Output: Best option for allocating resources W_b.

1. Initialize the whale population pop with P individuals represented as W_i (where $i = 1, 2, \ldots, P$). Set the current iteration t to 0 and define the maximum iteration T_{max}.
2. Select the best search agent $W_b(t)$ among the whales based on the fitness criterion.
3. **while** $t < T_{\text{max}}$ **do**
   (a) for each whale W_i in the population from 1 to P **do**
      i. Adjust parameters A, C, a, r, and $prob$.
      ii. **if** $prob < 0.5$ **then**
         A. **if** magnitude of A ($|A|$) is less than 1 **then**
            • Adjust D and W_i using Equations 4.1 and 4.2.
         B. **else**
            • W_{rand}, select a random whale.
            • Adjust D and W_i using Equations 4.4.
            Endif
      iii. **else**
         • Adjust D' and W_i using Equations 4.3.
         Endif
         EndFor
   (b) Adjust W_i if it goes beyond the search space.
   (c) Calculate the whale's W_i fitness.
   (d) Based on the fitness criterion, update W_b, the optimal search agent.
   (e) $t = t + 1$.
4. **end while**
5. Return W_b.

4.1.2 Earthworm Algorithm

The way that earthworms reproduce presents several optimization issues, however, the processes they take to reproduce can be made flawless by according to the following rules as shown in algorithm 5. Every earthworm can reproduce, and there are two different ways for each earthworm to do so.Every single earthworm-generated child possesses all the genetic material that is equal in length to the earthworms' parents.The earthworm that has the best fitness level

has direct next-generation permission and cannot be changed by operators. This can confirm that the earthworm population will not decrease as generations increase.

Algorithm 5: Earthworm algorithm

Initialization:

1. Convert t to 1 in the generation counter.
2. Set the MaxGen number of generations.
3. Decide on the NP population size.
4. Decide on the $nKEW$ number of earthworms to keep.
5. Decide on the similarity factor α.
6. Configure the proportional aspect β.
7. Establish the constant γ as 0.9.

Assessment of Fitness:

Adjust the aspect (fitness) of each earthworm to meet its location.

Main Loop: while the best solution is not achieved or $t <$ MaxGen do

1. All earthworms should be sorted by fitness value.
2. for $i = 1$ to NP do (loop over all earthworms).
   (a) x_{i1} children should be produced through reproduction 1.
   (b) Produce x_{i2} descendants by reproduction 2.
   End for
3. Execute a crossover operation on the offspring.
4. if $i > nKEW$ then
   (a) Choose N parents using the roulette wheel selection method.
   (b) Create M descendants.
   (c) Based on the produced offspring M, determine x_{i2}.
   else
   (a) Pick a single earthworm at random to serve as x_{i2}.
   (b) Update the earthworm's location.
   End if
5. for $j = nKEW + 1$ to NP do (loop over unkept earthworm individuals).
   (a) Execute the Cauchy mutation.
   End for
6. Calculate the population using the positions that have just undergone restructuring.
7. $t = t + 1$.

End while

Output:

The best solution should be presented.

4.2 Proposed Framework of WEOA (Whale Earthworm Optimization Algorithm)

The WEOA is implemented via the Load Balancer a shown in Figure 4.1. The tasks are delivered to the Load Balancer first,If the number of tasks is multiplied by the number of instructions exceeds the threshold, then load balancer communicates with the API gateway, which requests that the VM Controller construct extra worker virtual machines (Abualigah & Diabat, 2021). The API gateway obtains all of the machine parameters from the VM Controller and forwards them to the *"Load Balancer"*.

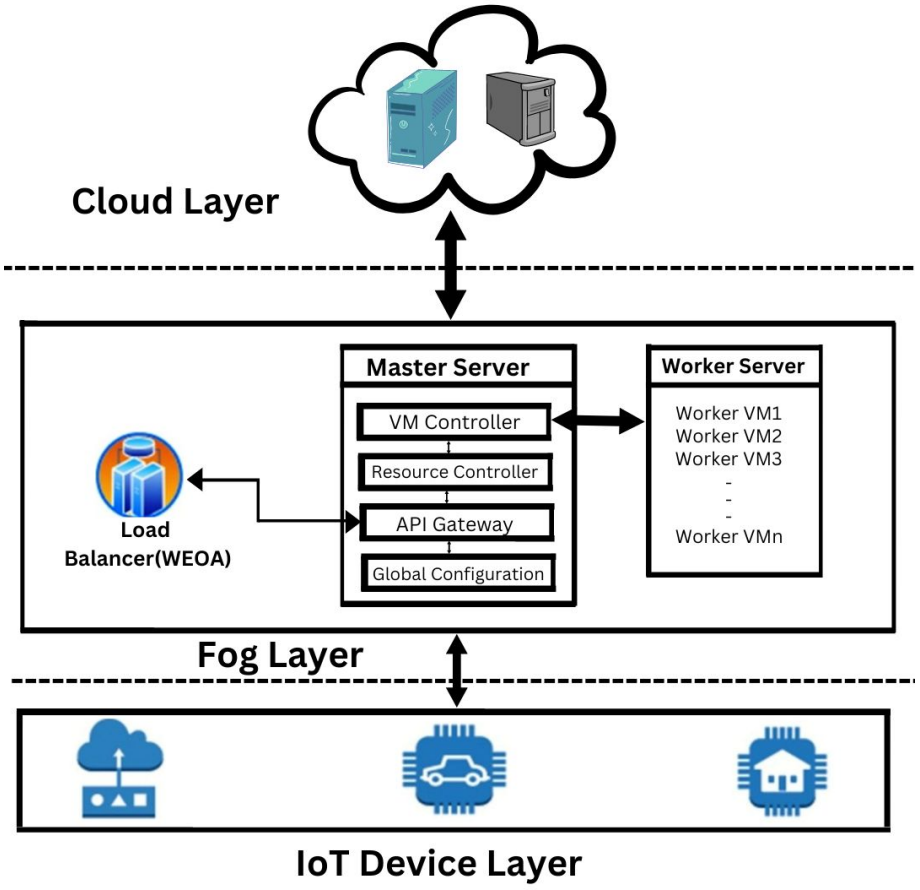


Figure 4.1: WEOA Framewok in Fog Environment

The *"Load Balancer"* then chooses the best machine using WEOA, tags tasks with machines to assign, and sends them to the *"Resource Controller"* over the API Gateway. The ideal machine's metadata is saved in the *"Global Configuration"* and can be utilized in subsequent cycles. The *"Resource Controller"* now

assigns all jobs to the worker VMs. The *"VM controller"* removes idle VMs to save power usage. The *"Resource controller"* forwards completed Tasks to the IoT device layer via Load Balancer. Depending on the size of the tasks, they will be routed to either the Fog or the Cloud devices. Depending on the computations, it may be moved from the Fog to the cloud or from the cloud to the Fog during task processing.

4.2.1 Proposed solution to problem and Implementation

Figure 4.1 depicts the task scheduling system, where Task T=(t1,t2,....tn) represented the jobs received from IoT devices. FN1, FN2,....FNn are the Fog nodes, and CN1.....CNn) are the cloud nodes, which are in charge of task execution. However, in the Fog-Cloud scenario, there is an issue with the efficient assignment of tasks to these nodes. The overall number of server nodes in the system is the sum of Fog and Cloud nodes. All of these nodes are heterogeneous in terms of bandwidth (Nb) and processing speed (Nps).

The Fog nodes process the tasks first, followed by the cloud nodes. The primary motivation for this research is to determine how to efficiently assign jobs to these nodes in order to minimize cost, time, and reaction time. When allocating jobs to nodes, our load balancer algorithm will take into account all of these limits.

The goal of this work is to reduce costs, makespan, and response time. These three parameters are taken into account in this work since most applications, such as self-driving cars, healthcare, gaming, and media, require the quickest response time, the largest throughput, and the lowest operational cost. As a result, these characteristics are crucial in any Fog-Cloud-based computing system.

Cost

There is always a cost associated with the work accomplished by nodes either by Fog nodes or by cloud. Task computation costs and communication costs are always there in a system. Computation cost can be computed by the cost of memory, processing, and bandwidth occupied by the nodes and Communication cost is the cost of time consumption in allocating resources, it can be computed by network latency, time required for execution of resource management algorithm and number of devices requesting resources. Total-Cost in network can

be computed by combination of Computation cost and communication cost as shown in equation 4.5.

$$Total - Cost = \psi_{\text{Comp-cost}} + \psi_{\text{Comm-cost}} \tag{4.5}$$

Computation Cost

Computation cost can be computed by the cost of memory, processing, and bandwidth occupied by the nodes.

$$\psi_{\text{Comp}-\text{cost}} = \psi_{\text{cost}}(m) + \psi_{\text{cost}}(p) + \psi_{\text{cost}}(b) \tag{4.6}$$

The Computation Cost of a system may be computed by the cost of memory utilization, the cost of CPU, and the cost of Bandwidth as mentioned in above equation 4.6.

$$\psi_{\text{cost}}(m) = \sum_{i=1}^{m} c_i^m \tag{4.7}$$

The cost of memory taken by nodes Ni during job completion is described in equation 4.7. It is calculated using the cumulative memory used by Fog nodes and the cloud.

$$\psi_{\text{cost}}(p) = \sum_{j=1}^{n} c_j^p \tag{4.8}$$

Similarly, the above equation 4.8 describes the cost of using the CPU by nodes Ni during task processing, and it also includes processing during the wait time. Total CPU cost is the sum of cloud and Fog costs, as shown in equation 4.9.

$$\text{Total}_{\text{CPU}}^{\text{Cost}} = C_{\text{CPU}}^{\text{cloud}} \cup C_{\text{CPU}}^{\text{Fog}} \tag{4.9}$$

The cost of bandwidth must also be factored into the overall system cost. Bandwidth utilization on Fog nodes and in the cloud may be the same. The total bandwidth utilized by both Fog and cloud during the system's working period can be used to calculate bandwidth cost, as shown in equation 4.10.

$$\psi_{\text{cost}}(b) = \sum_{k=1}^{l} c_k^b \tag{4.10}$$

Communication Cost

Communication cost is the cost of time consumption in allocating resources, it can be computed by network latency in a system, time required for execution of

resource management algorithm and number of devices requesting resources.

$$\psi_{\text{Comm-cost}} = T_{\text{net-latency}} + T_{\text{algo-exec}} + N_{\text{Dev}} \tag{4.11}$$

Based on the above equation 4.11, The network latency $T_{\text{net-latency}}$ will affect the time of resource allocation between the Fog nodes and the devices. Reduced latency leads to faster resource allocation and it can be measure in seconds(s). The efficiency of the Fog environment resource management algorithm can have a substantial impact on time consumption $T_{\text{algo-exec}}$ and it can be measure in seconds. A well-optimized algorithm can quickly discover available resources and more efficiently allocate them. The total number of devices N_{Dev} connected to the Fog environment can affect the time it takes to allocate resources. Handling additional devices may necessitate more time for resource allocation and administration.

Based on the equation 4.6 and 4.11, total cost in a system can be computed as per equation 4.12.

$$\text{Total-Cost} = (\psi_{\text{cost}(m)} + \psi_{\text{cost}(p)} + \psi_{\text{cost}(b)}) + (T_{\text{net-latency}} + T_{\text{algo-exec}} + N_{\text{Dev}}) \tag{4.12}$$

Makespan Makespan can be calculated by subtracting the time of job completion from the time of task start. It is the time it takes to complete a task from start to end. Makespan is calculated as per equation 4.13

$$\text{MS} = \min\left\{F_{t_i}^{\text{time}} - S_{t_i}^{\text{time}}\right\}, \quad \text{where } t_i \in T \tag{4.13}$$

Here $F_{t_i}^{\text{time}}$ is the finishing time of the task, and $S_{t_i}^{\text{time}}$ is the starting time of the task. Makespan must be minimum for efficient task scheduling.

Response-Time The machine's response time to any enquiry is defined as response time. Response times can be estimated for n concurrent user processes and m requests.

$$RT = \frac{n}{m} + W_t \times \text{time} \tag{4.14}$$

In the preceding equation 4.14, numerous questions will be raised by many users n, and all of these questions m will be answered by machines at any given time. Waiting time between two jobs might be included to calculate response time accurately. To process any inquiry, a system may have a long response time.

The average reaction time is the time it takes machines to complete n tasks, where Ti=t1,t2,...tn. The average response time can be calculated using equation 4.15.

$$ART = \frac{\sum_{i=1}^{m} T_i}{n} \tag{4.15}$$

Ti is the machine's individual time for task completion, and n is the number of tasks completed.

Objective-Function

According to the preceding discussion, the objective function of our task, as stated in equation 4.16, is to minimize cost, time, and reaction time.

$$\textbf{Objective-function} = We1 * \psi_{\text{cost}} + We2 * MS + We3 * RT \tag{4.16}$$

Where We1, We2, and We3 are the relevant metric weights. The weightage of cost, time, and reaction time must be equal to one, as described in equation 4.17.

$$\sum_{i=1}^{3} We_i = 1 \tag{4.17}$$

4.2.2 Algorithm for proposed WEOA

This section discusses the algorithm we suggest, Algorithm 6. WEOA is a mix of the whale optimization and Earthworm optimization algorithms. The advantages of both optimization techniques are used to improve the system's efficiency.For reproduction, the suggested technique employs the earthworm optimization technique, with only reproduction 2 from the classic earthworm algorithm being used. The earthworm algorithm's cross-over operation aids in the replication of numerous machines in our proposed approach, and the whale optimization algorithm is employed for encircling and hunting the prey (tasks). The proposed approach aids in the optimization of the exploration phase and convergence speed, which aids in the discovery of optimal resource management

solutions in the Fog-Cloud system.

Algorithm 6: The WEOA pseudo-code

Input: $Max_{\text{iteration}}, Threshold_{\text{Task-Size}}$.;
Output: Efficient Depict of tasks by reducing ψcost, MS, RT.;
Procedure: Start;
1. Initialize population W of whales and Earthworms. // Count of Machine M and Number of Tasks.;
2. Evaluate the Fitness Function.;
3.**While** $current_{\text{itr}} \leq Max_{\text{iteration}}$ **do**;
4. **if** prey (Task-size) > $Task_{\text{Threshold}}$ // Check Task-Size based on the instructions.;
then;
5. Evaluated the fitness of Whales. ;
6. Cross-over reproduction from fitness and append to (W) as equation 4.18. // Initializing heterogenous machines from fitness.;
7. **elif** $Encircling_{\text{prob}}$ > 0.5;
then;
8. Update the whale position using encircling equation 4.19. //Update global configuration with the optimal machine.;
9. **else do**;
10. update the whales position randomly. //Update global configuration with optimal machine;
11. **End if**;
12. **End if**;
13. **End-While**;
14. Search for an optimal Machine from $global_{\text{config}}$;
15. Optimal Machine Found.;
16. **End Procedure**;

Description of an Algorithm

1. **Initialization** The whales' placement represents a plausible answer to an optimization challenge. The primary goal of this work is to compare efficient mapping between tasks and machines. The whales and earthworms are initialized in step 1 of this procedure. The abundance of earthworms aids in whale reproduction. A random maximum number of whale positions aids in the search for the best-optimized solution in the search space.

2. **Iteration Process and Fitness Function** The algorithms' next steps deal with the iteration process and fitness function, which determine which search agent is the best in the search space. The maximum iteration (Max-iteration) value will continue the iteration process. The job size will be determined in each iteration depending on the task size threshold value. If the task size exceeds the threshold value, the machine's fitness will be evaluated using the supplied equation. If the machine's fitness is no longer sufficient, the reproduction strategy will be selected using the earthworm optimization process. Optimization algorithm stages exploration and exploitation will be done during each iteration process.

3. **Exploration phase based on Earthworm strategy:** The Earthworm optimization approach is used in the exploration phase of this algorithm. Machine reproduction or replication will proceed according to equation 4.18:

$$Xy(i) = \begin{cases} Xa(i), & 1 \leq i < k \\ Xb(i-k), & k < i \leq m \end{cases} \tag{4.18}$$

Let A and B be the two machines; we will reproduce new machine Y using A and B. Let Xa represent machine A's feature vector, Xb represent machine B's feature vector, and Xy represent machine Y's feature vector. Because we are using crossover reproduction (EOA), the offspring will have some random characteristics from A and some from B. The above equation demonstrates how the traits in Y are inherited from A and B.

As per the above equation, the "i^{th}" feature of Y will be an "i^{th}" feature from the parent A, if $1 \leq i < k$, and "i^{th}" feature of Y will be the "$(i-k)^{\text{th}}$" feature from B if $k < i \leq m$. where "k" is called the cross-over point and is a random integer between 1 & n and "n" is the length of feature vectors of A and B.

As a result, the progeny Y will be a hybrid species with random traits of A and B, indicating cross-over reproduction.

Note*: A vector is a single column or row matrix. thus, a feature vector is a single column or row matrix containing features as its elements.

4. **Encircling the prey:** Whale encircling is relied on spiral encircling; whales are positioned in search space based on the encircling approach or by random position based on the position of other whales. If the likelihood of encirclement exceeds 0.5, the position of the whales is altered as shown in equation 4.19:

$$\begin{cases} X(t+1) = X(t) + A \cdot \cos(a \cdot t) \cdot \cos(b \cdot t) \\ Y(t+1) = Y(t) + A \cdot \cos(a \cdot t) \cdot \sin(b \cdot t) \\ Z(t+1) = Z(t) + A \cdot \sin(a \cdot t) \\ \vec{X}(t+1) = X(t+1)\hat{\mathbf{i}} + Y(t+1)\hat{\mathbf{j}} + Z(t+1)\hat{\mathbf{k}} \end{cases} \tag{4.19}$$

The encircling motion of the whales is represented by the preceding equation, and the functions x(t), y(t), and z(t) give the x, y, and z coordinates of the whale in 3-dimensional space, respectively. The amplitude of the whale's spiral movement is represented by A in the above equation, while a and b are two whale-specific angles.

The value of t, which varies from 0 to 1, reflects the progression of the spiral movement over time.

The final equation is used to convert vectors from coordinates returned by the functions. As a result, X is the whale's position vector.

5. **The exploitation phase is based on the Whale attacking mechanism.** The exploitation phase aids in the hunt for prey; in our proposed approach, the exploitation phase is carried out in accordance with the whale optimization algorithm, as shown in equation 4.20:

$$\vec{X}(t+1) = m \cdot (\vec{P} - \vec{X}(t)) + \vec{X}(t) \tag{4.20}$$

Let P be the prey's location vector and X(t) be the whale's position vector at time t. The above equation represents the whale's attacking character. The movement factor "m" here is between [0, 1]. It is determined by things such as the machine's specifications or the behaviour of other whales in the population.

The operation will be continued till the maximum number of iterations is reached. The ideal solution will be found from the global configuration after the maximum number of iterations are completed. If the ultimate ideal solution from the global configuration is determined, the procedure will be completed.

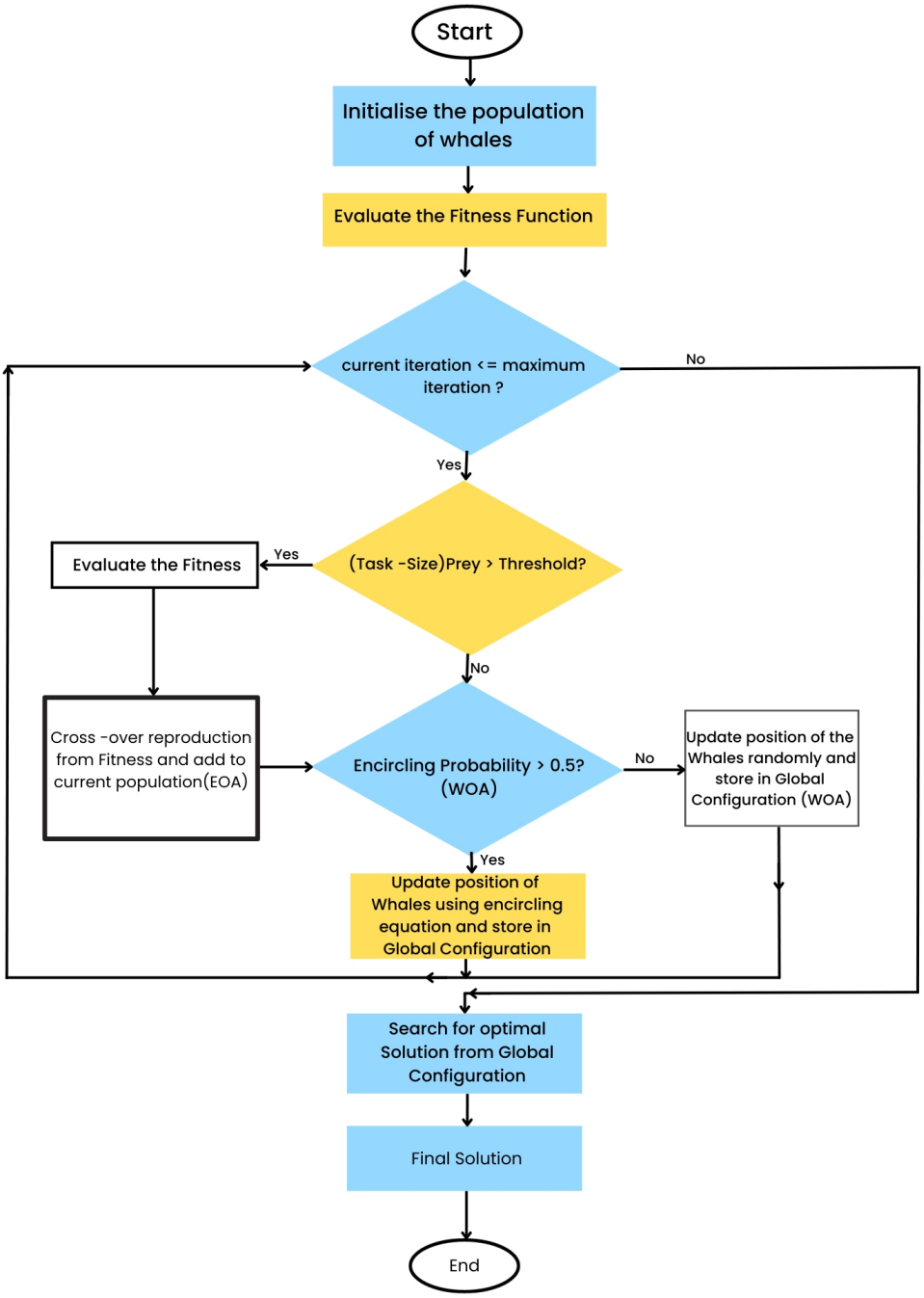


Figure 4.2: Methodology of WEOA Framewok

Figure 4.2 depicts the proposed WEOA process. The flowchart begins with the initialization phases, which include the initialization of whales and earthworms. The method is continued until an optimal solution is identified based on the iteration number specified. The fitness function of the machines is derived using equation 4.16, which takes into account cost, makespan, and response time. The entire system is employed in vehicular Fog computing situations where automatic automobiles are operating on roadways and real-time data without delay is required for automatic vehicles to run. According to the Performance improvement rate stated in the result section for the vehicular Fog computing environment, response-time is weighted (We1) at (0.60), makespan is weighted (We2) at (0.20), and cost is weighted (We3) at (0.20). Response-time weightage is substantially higher since data on traffic bottlenecks, humps, and when to apply breaks are required on a high priority without delay for making decisions and avoiding an accident.

4.2.3 Explaining example for the proposed technique

As illustrated in Figure 4.3 current state-of-the-art (Alizadeh et al., 2020), IoT-Fog-Cloud environment designs include VMs that are identified as Fog devices with static functionalities. As demonstrated in Figure 4.3, some VMs are idle i.e., those are without assigned jobs for numerous tasks. As a result, Fog devices 2 and 4 are labelled as idle devices, resulting in unnecessary hardware utilization. Furthermore, tasks having instructions above the threshold value must be delivered straight to the cloud layer for computation (Rahbari & Nickray, 2019; Shen et al., 2017). For heterogeneous architecture, there are chances that some tasks are left unallocated despite the resources available for the allocation, and the main reason behind this is the atomic nature of the tasks. The existing state of the art has some cons such as inefficient allocation of resources, improper distribution of resources among different VMs, and inability to allocate tasks to machines because of the atomic nature of the task.(Naas, Lemarchand, Raipin, & Boukhobza, 2021)

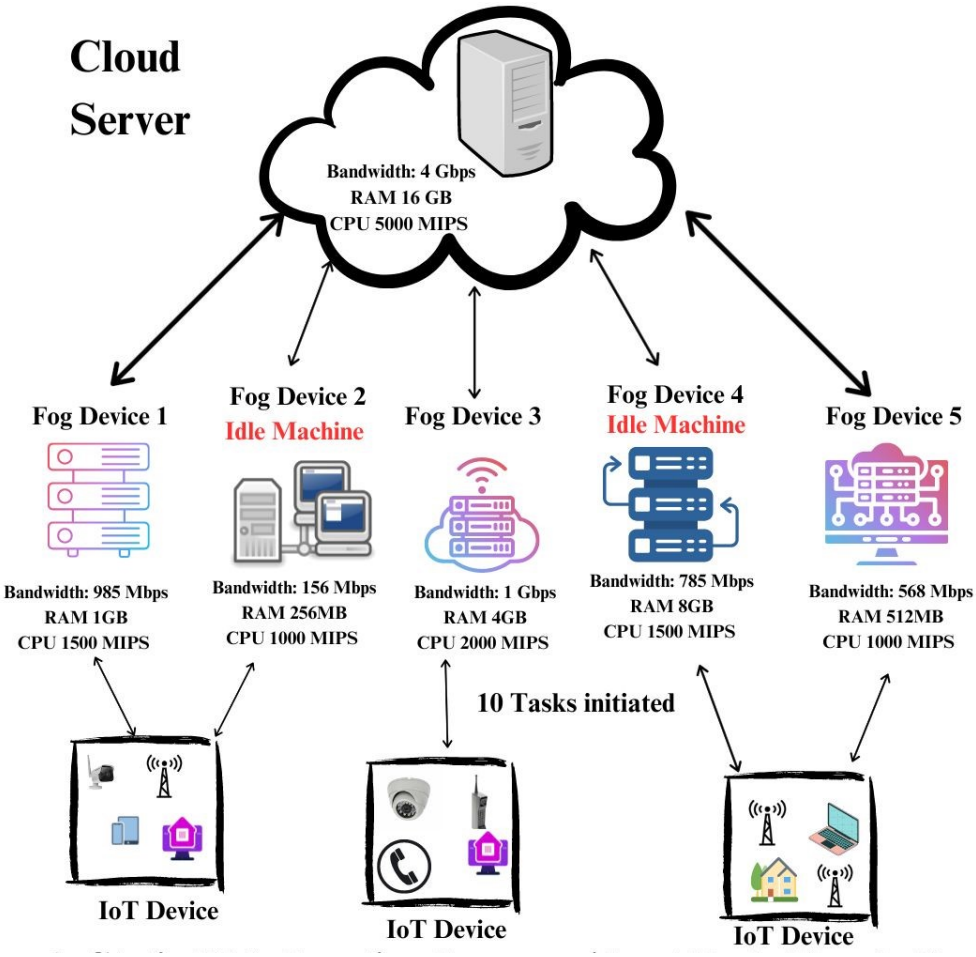


Figure 4.3: Static VM allocation process without Task-Threshold

As shown in Figures 4.4 and 4.5, the proposed approach uses WEOA for the Cloud-Fog architecture. The WEOA is a hybrid genetic algorithm that has the ability of reproduction (Earthworm cross-over reproduction) to reproduce the VMs as per threshold, and the whale optimization algorithm helps in whale encircling and attacking the prey. Initially, a random number of machines are initialized, and their fitness value is calculated. The load balancer iterates over the tasks queue to check if the task queue is greater than the threshold. Thus, the strength of the Earthworm's cross-over reproduction can eradicate the issues caused by the atomic nature of the Task. The proposed architecture has a Threshold value for the task size that decides whether the VMs need to reproduce or not. If the task size exceeds the Threshold, then the cross-over reproduction takes place and new VMs are reproduced.

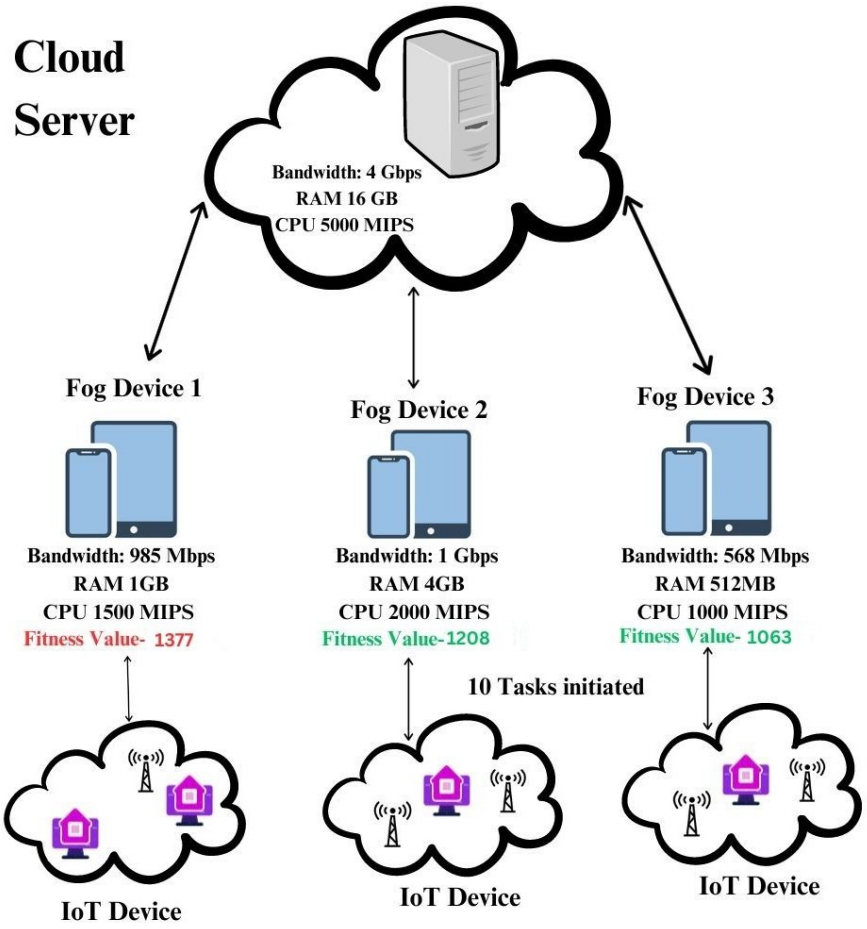


Figure 4.4: Dynamic VM allocation process using WEOA before Machine Replication

The parents for the machine to be reproduced are selected based on the least fitness value and the one with the second least fitness value as done in Figure 4.4 Machine 2 and Machine 3 have the least fitness values 1208 and 1063. Then replication process will be done through these machines as shown in Figure 4.5. With each iteration, the fit-ness of the machine is calculated, and the Whale's encircling phenomenon plays its role. The VMs iterate over the task queue again, taking into consideration, the newly reproduced VMs, the task is allocated dynamically. Due to the dynamic allocation of the tasks, the problems caused by the atomic nature of tasks (tasks are independent and cannot be di-vided into chunks) such as improper allocation of resources are coped up. For the case, where the number of tasks is less than the running VMs, VMs with high fitness value are shut down and tasks are only allocated to the machines with low fitness value.

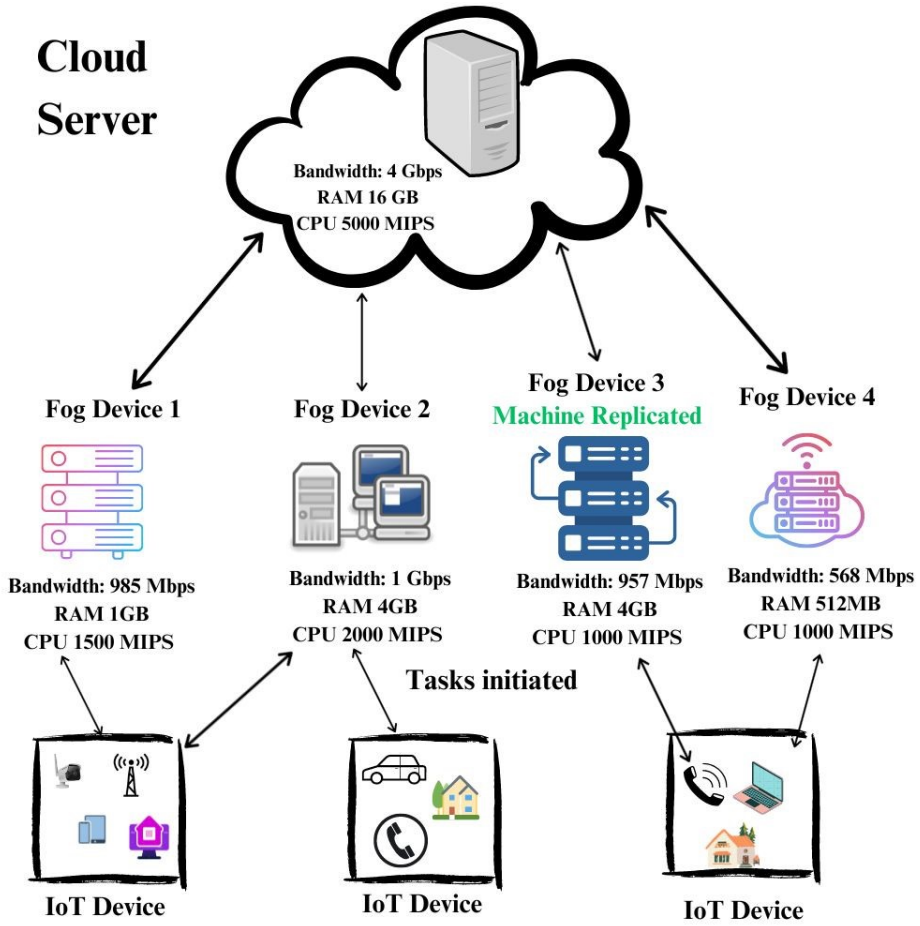


Figure 4.5: Dynamic VM allocation process using WEOA after Machine Replication

This dynamic nature of the architecture helps us to reduce the unwanted usage are resources that were being used by the idle machines and thus, saving power consumption. Thus, the proposed architecture has two major processes, the first one is the reproduction of machines as per threshold value and the second one is the allocation of resources by iterating over the task queue to allocate tasks selectively.

4.3 Experimental Evaluation

In this segment, we are discussing regarding experimental set-up and comparison contrasting with traditional techniques. All experiments are performed on the iFogSim toolkit(H. Gupta et al., 2017) with system setup core i7, Windows 10 OS, and 16GB of RAM.

4.3.1 Experiment setup and dataset statistics

The data-workflows for performance checking of the proposed approach are brought out from real-world data sets HPC2N and CEA-CURIE https://www.cs.huji.ac.il/labs/parallel/workload. The execution traces bring out from the processing of coinciding HPC tasks exist in these workload logs. Table 4.2 illustrates the real workloads used in this work. In a real scenario, HPC2N and CEA-Curie have many workloads, but in this work, we have taken only 10 workloads.

Table 4.2: Illustration of real workloads

Workload Log	**Parallel Tasks**	**CPUs**	**Users**	**Filename**
HPC2N	202871	240	257	HPC2N-2002-2.2-cln.swf
CEA Curie	312826	93312	582	CEA-Curie-2011-2.1-cln.swf

For simulation setup, both Fog and cloud nodes are taken for processing. Evaluation of the experimental setup has been done on parameter Cost, makespan, and response-time. Each machine on Fog and Cloud has some level of bandwidth, processing speed, and RAM. Fog nodes have a lower range of bandwidth, CPU frequencies, and RAM utilization in comparison to the cloud. Cost is considered with a unit of (Grid$). Table 4.3 describes the configuration details of the Cloud and Fog scenario.

Table 4.3: Configuration of Cloud and Fog scenarios.

Parameters	**Fog**	**Cloud**	**Units**
Processing Speed	[1000:2000]	[3000:5000]	MIPS
Bandwidth	[128:1024]	[512:4096]	Mbps
RAM	[250:5000]	[5000:20000]	MB
Cost	[0.2:0.5]	[0.6:1.0]	G$
VMs Numbers	[15,20,35]	[10,15,20]	VM

4.3.2 Simulation results

For evaluation of simulation results proposed approach is compared with an existing technique that was implemented using optimization algorithms like whale optimization, earthworm, cuckoo search, etc. We have compared our proposed approach with five existing techniques including h-DEWOA (hybrid-differential-evolution-enabled whale-optimization algorithm)(Chhabra, Sahana, Sani, Mohammadzadeh, & Omar, 2022), cuckoo-search- differential algorithm (CSDEO)(Chhabra et al., 2021), Cuckoo-search-particle-swarm-optimization algorithm (CSPSO)(Chhabra, Singh, & Kahlon, 2020), blacklist matrix-based-multi-objective algorithm (BLEMO) (Vila, Guirado, Lerida, & Cores, 2019), and EEOA (Electric-earthworm-optimization algorithm) (Kumar & Karri, 2023). All comparisons are made for 30 iterations for 10 workloads from given HPC2N and CEA-CURIE datasets.

Results for CEA-Curie workload

Figure 4.6 is shown comparison results on performance metric makespan for the proposed approach with all mentioned five techniques. As per comparison results CSPSO failed to perform better among all techniques, the value of makespan for CSPSO is much higher than all the techniques. In the h-DEWOA technique differential evolution technique is integrated with the whale algorithm, as the whale algorithm is having the problem of getting remain in local optima and low convergence speed. h-DEWOA also doesn't have fair makespan results. The proposed approach outperforms all mentioned techniques on performance metric makespan, the approach is getting the benefit of dynamic machine allocation, and load balancer approach, which continuously replicates the machine in case if tasks with higher threshold values arrived during iterations.

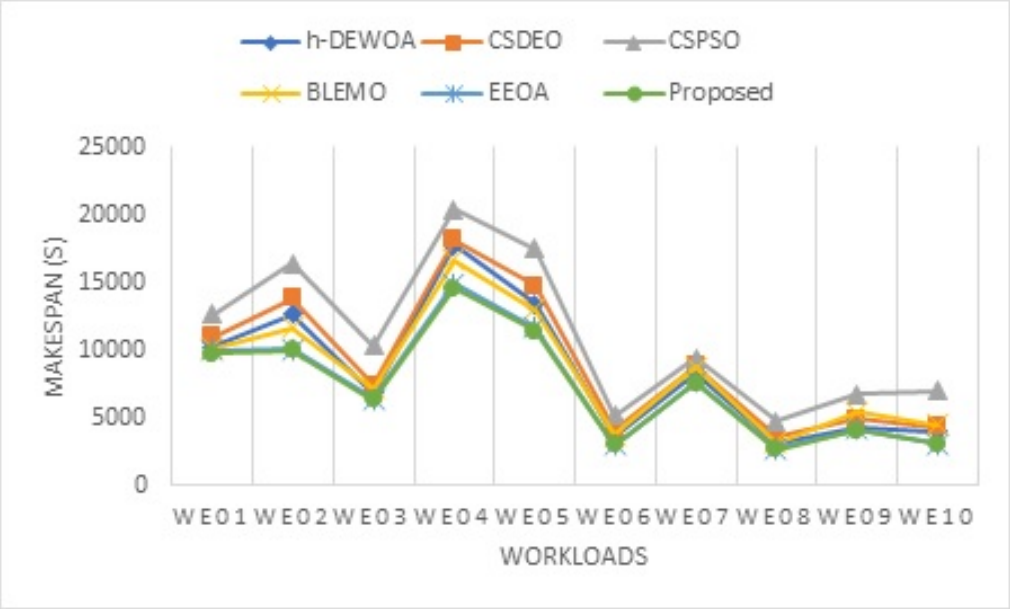


Figure 4.6: Best Makespan for CEA-Curie workload

Performance improvement rate is calculated to determine the % of improvement of the proposed approach with all mentioned techniques as shown in Table 4.4, performance is calculated by the equation 4.21

$$\frac{p_{\text{pre}} - p_{\text{pro}}}{p_{\text{pro}}} \tag{4.21}$$

Here p_{pre} is the makespan value of the previous approach, and ppro is the makespan value of the proposed approach. As shown in Table 4.4, the proposed approach successfully gets performance over the techniques. For workload, WE01 best case makespan value of EEOA is 9987.45, and for the proposed approach it's 9788.07, then as per the PIR equation $\frac{9987.45-9788.07}{9788.07} \approx 2.04$ %. That's why a 2% improvement is noticed by the proposed approach over EEOA.

Table 4.4: PIR (%) of the proposed approach with all existing techniques

Workloads	h-DEWOA	CSDEO	CSPSO	BLEMO	EEOA
WE01	4.98	12.07	29.50	4.21	2.04
WE02	26.83	38.54	63.81	17.03	1.05
WE03	7.82	17.50	62.80	13.68	1.94
WE04	21.78	24.61	40.16	14.14	2.38
WE05	17.80	29.58	53.86	14.19	2.49
WE06	20.92	40.36	73.97	26.87	1.75
WE07	8.55	15.97	23.43	15.03	1.15
WE08	18.28	34.71	76.54	17.25	1.38
WE09	5.33	20.53	65.84	32.03	2.69
WE10	30.08	42.46	127.87	46.76	1.83

Figure 4.7: Cost comparison for CEA-Curie workload

Figure 4.7 shows the cost comparison on CEA-Curie workload, as per representation CSPSO has a higher cost in the system. BLEMO, h-DEWOA, and CSDEO have approx the same cost for workload WE01-WE08. Our proposed approach outperforms the EEOA technique and has a performance improvement

of 2% on the CEA-curie workload. The proposed technique performs efficiently with lower cost value because VMs are allocating effectively. The load balancer and resource controller are managing the network together. If the load balancer raises some inquiry through the API gateway regarding task mapping, the resource controller will check the available resources with each machine. No idle machine will remain in the network helps in reducing the cost of the proposed approach.

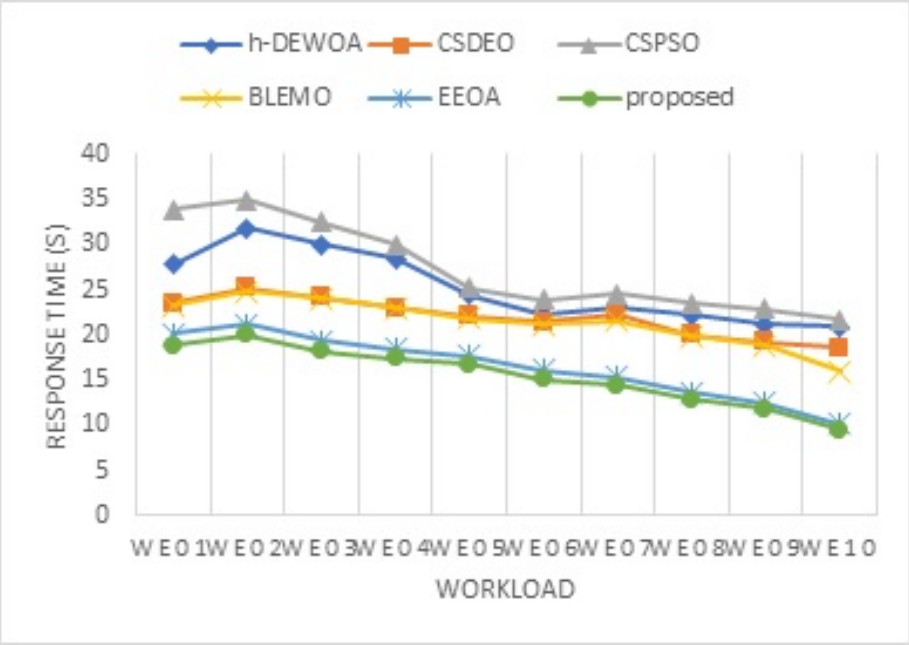


Figure 4.8: Response-time comparison for CEA-Curie workload

Figure 4.8 represents the response-time comparison with existing techniques on the CEA-Curie workload. Response time is calculated using the communication time with think time in the system. Delay in the CSPSO technique is higher, that's why RT of the CSPSO technique is more in comparison with all techniques. Task mapping of CSDEO and BLEMO is almost the same. Both technique is vector based where a vector is maintained for mapping the tasks with VMs. The proposed approach has low RT in comparison to existing techniques. The whale optimization exploitation phase helps in responding to tasks fastly. In contrast, our algorithm has responded effectively on all workloads. the proposed approach successfully gets performance over the techniques. For workload, the WE01 RT value of EEOA is 20.14, and for the proposed approach it's 18.89, then as per the PIR equation $\frac{20.14-18.98}{18.98} \approx 6.11$ %. That's why a 6% improvement is noticed by the proposed approach over EEOA.

Results for HPC2N Workload Figure 4.9 shows comparison results on per-

formance metric makespan for the proposed approach with all mentioned techniques. As noticed in Figure 4.9, the CSPSO technique is performing badly on the HPC2N workload for the Performance metric makespan. The task mapping technique of CSPSO is unable to perform on the high-performance network. h-DEWOA and BLEMO approaches is having the same performance on 10 workloads under 30 iterations. Our proposed approach performed better and outperforms other techniques. VMs in the proposed approach is managed properly by the VM controller and resource controller even for the High-performance network dataset. Here Exploitation phase, which is running under the control of the whale optimization algorithm is responding well.

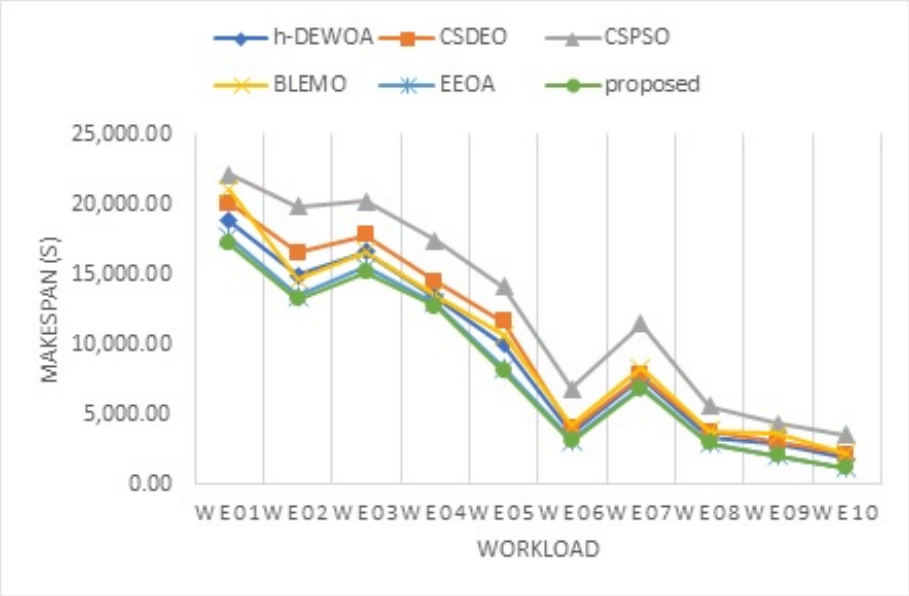


Figure 4.9: Best Makespan for HPC2N Workload

Performance improvement rate is also calculated for the HPC2N workload for the proposed technique with other techniques using equation 4.21 as shown in Table 4.5. For workload, WE01 best case makespan value of EEOA is 17612.19, and for the proposed approach it's 17225.72, then as per the PIR equation $\frac{17612.19-17225.72}{17225.72} \approx 2.24\ \%$. That's why a 2% improvement is noticed by the proposed approach over EEOA.

Table 4.5: PIR (%) of the proposed approach with all existing techniques

Workloads	h-DEWOA	CSDEO	CSPSO	BLEMO	EEOA
WE01	6.75	13.76	25.53	19.66	2.25
WE02	11.16	22.83	47.361	8.99	1.45
WE03	7.32	14.81	30.45	7.49	2.22
WE04	3.53	12.42	35.17	4.81	1.58
WE05	19.73	39.48	69.51	28.71	2.99
WE06	15.48	25.24	78.33	31.58	2.12
WE07	7.80	11.31	62.77	18.25	2.25
WE08	11.71	26.43	78.17	28.53	1.93
WE09	41.37	43.13	79.44	73.08	2.40
WE10	54.80	75.91	80.66	78.36	1.88

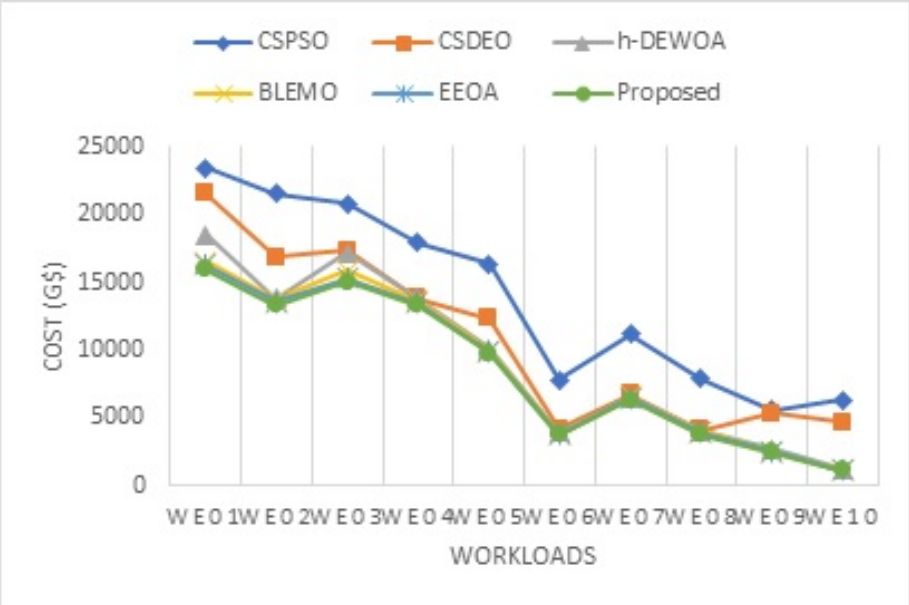


Figure 4.10: Cost comparison for HPC2N workload

Figure 4.10 shows the comparison of the total cost for the HPC2N workload, here cost is measured in G$. The cost factor is determined by the parameter's bandwidth, CPU, and RAM. As per 4.10, the cost of CSPSO is much high, and

the cost of BLEMO & h-DEWOA is equally approached. The cost for workloads WE04 and WE05 has been raised for all techniques. The proposed technique has less cost in comparison to EEOA and other techniques. 2% of performance is noticed for the proposed approach over the EEOA technique. As per the proposed approach cost of using VMs is low because VMs are generated dynamically as per the requirements. but in the case of the CSPSO technique, all VMs are initialized in starting without seeing the load of the network. That's why the cost of the CSPSO technique is much high as compared to other techniques.

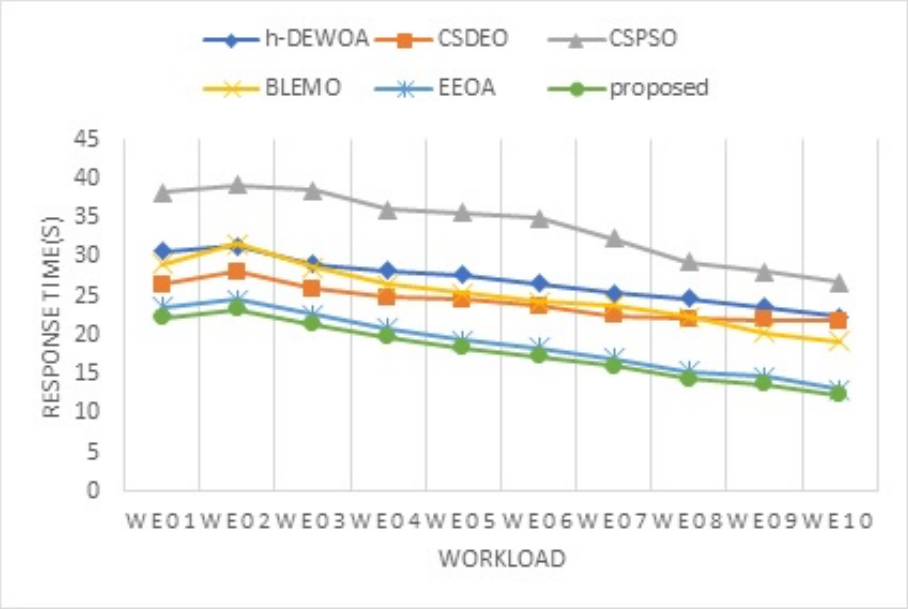


Figure 4.11: Response-time comparison for HPC2N workload

Figure 4.11 represents the response-time comparison with existing techniques on HPC2N workload. Similarly, for the HPC2N workload, the delay in the CSPSO technique is higher, that's why the RT of the CSPSO technique is more in comparison with all techniques. The proposed approach has low RT in comparison to existing techniques. In contrast, our algorithm has responded effectively on all workloads. the proposed approach successfully gets performance over the techniques. For workload, the WE01 RT value of EEOA is 23.58, and for the proposed approach it's 22.18, then as per the PIR equation. $\frac{23.58-22.18}{22.18} \approx 6.31$ %. Due to this, a 6% improvement is noticed in the proposed approach over EEOA.

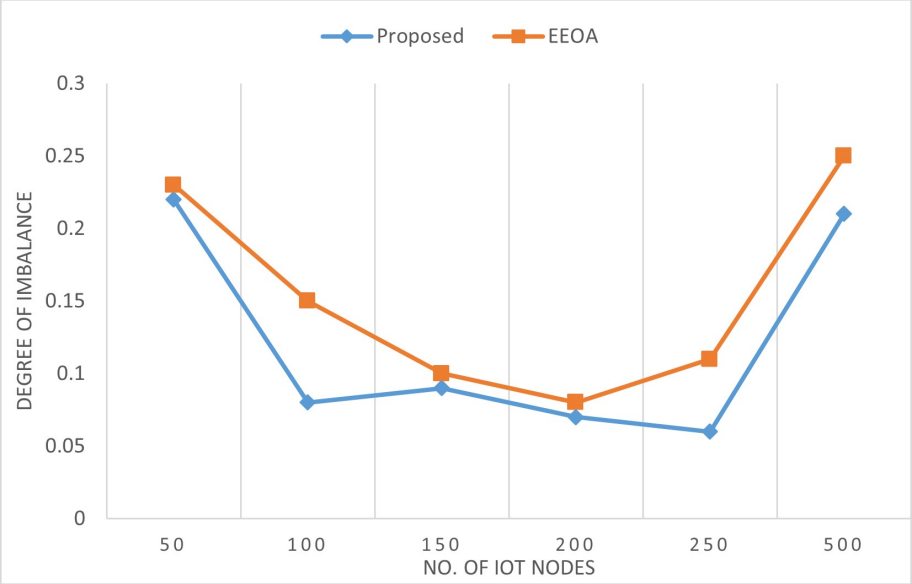


Figure 4.12: Objective Function value throughout a single run of the proposed algorithm for CEA-Curie workload.

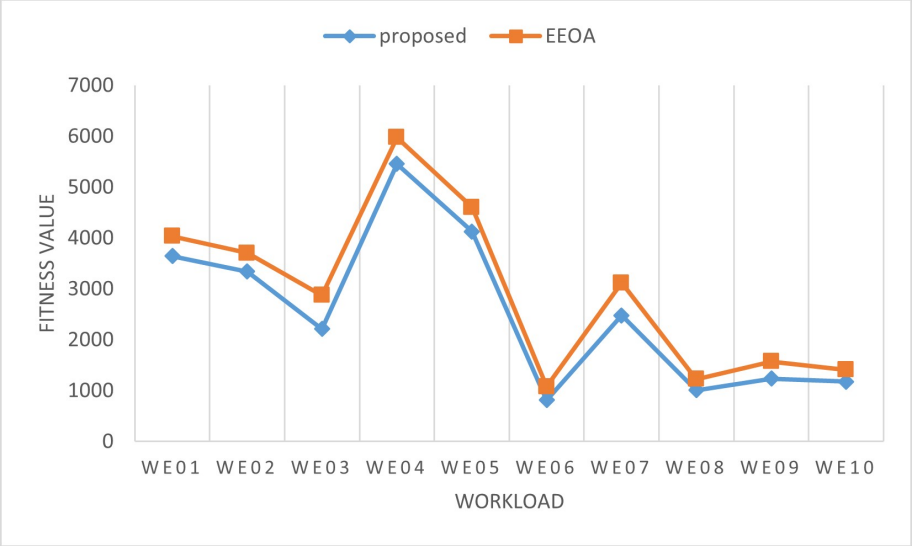


Figure 4.13: Objective Function value throughout a single run of the proposed algorithm for HPC2N workload.

Figures 4.12 and 4.13 display the objective function values throughout a single run iteration for both the CEA-Curie and HPC2N algorithms. As per Equation 4.16 , the fitness value for both the proposed and EEOA algorithms is calculated, considering the values of We1 and We2, which are 0.2 and 0.6, respectively. The suggested WEOA algorithm exhibits quicker convergence compared to the EEOA algorithm. The results indicate that, when contrasted with the EEOA algorithm, the suggested WEOA algorithm effectively explored the solution

space and reached the optimal value within a comparable timeframe.

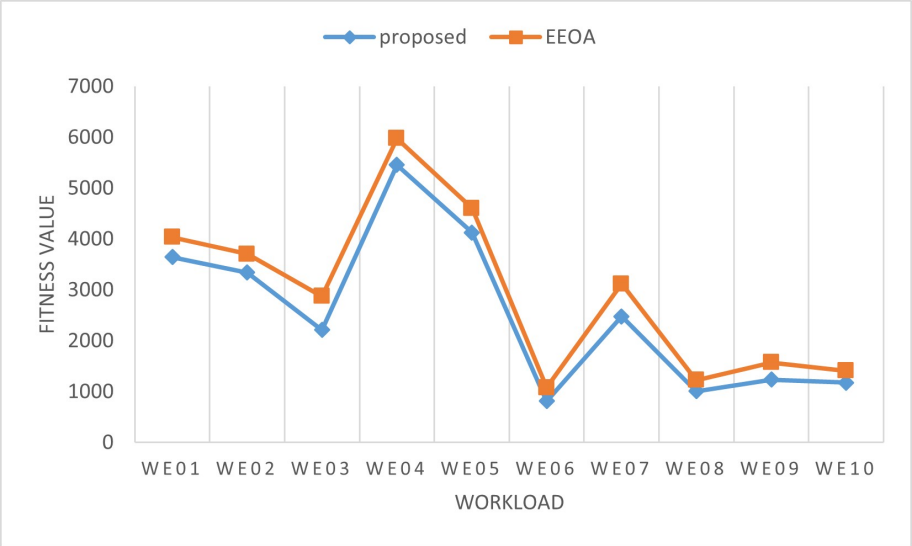


Figure 4.14: Degree of imbalance of the offloaded the tasks.

Figure 4.14 depicts the degree of imbalance for the EEOA algorithm and the proposed approach using the same previous parameter settings as the number of IoT nodes increases. The graph demonstrates that the suggested WEOA algorithm maintains less values. This suggests that the proposed approach successfully distributes the workload across the Fog nodes.

Discussion

The positive results of this study demonstrate the efficiency of the suggested WEOA algorithm in optimizing resource management in a Cloud-Fog scenario. The algorithm creates a balance between exploration and exploitation by combining the strengths of the Earthworm and Whale optimization algorithms, resulting in enhanced job allocation and overall system performance. The findings of this study have practical relevance for service providers operating in Cloud-Fog environments that generate enormous amounts of data from IoT devices. These providers can improve resource utilization, reduce costs, improve response times, and achieve higher task completion rates by applying the WEOA algorithm.

4.4 Statistical validation for WEOA

Table 4.6 shows the results of an ANOVA statistical analysis of two factors without replication on the degree of imbalance. The proposed algorithms outperform traditional algorithms in terms of analytical efficiency. In statistical analysis, a p-value less than 0.05 and an F crit value less than F demonstrate the capability

of the suggested approach. According to this analysis, the proposed approach is superior for handling optimization challenges on the IoT-Fog-Cloud system.

Table 4.6: Anova Statistical Analysis: Two Factor Without Replication For Degree Of Imbalance

Source of Variation	**SS**	**df**	**MS**	**F**	**P-value**	**F crit**
Rows	**0.050375**	**5**	**0.010075**	**30.68528**	**0.000929**	**5.050329**
Columns	**0.003008**	**1**	**0.003008**	**9.162437**	**0.029183**	**6.607891**
Error	**0.001642**	**5**	**0.000328**			
Total	0.055025	11				

Chapter 5

Conclusion and Future Direction

The findings of this study have practical relevance for service providers operating in Cloud-Fog environments that generate enormous amounts of data from IoT devices. These providers can improve resource utilization, reduce costs, improve response times, throughput, and achieve higher task completion rates by applying the WEOA,and IGWOA algorithms. The successful outcomes of this study demonstrate the efficiency of the suggested WEOA,and IGWOA in optimizing resource management in a Cloud-Fog scenario. The algorithms creates a balance between exploration and exploitation , resulting in enhanced job allocation and overall system performance.

When WEOA compared to EEOA, the results show a 6% improvement in reaction time, 2% in cost, and 2% in makespan. Furthermore, when compared to other systems such as h-DEWOA, CSDEO, CSPSO, and BLEMO, it demonstrates remarkable improvement of up to 82% in reaction time, 75% in cost, and 80% in makespan.

The proposed IGWOA approach is contrasted with three other well-known optimization algorithms: AEOSSA, HHO (Harris Hawks Optimization), and PSO (Particle Swarm Optimization), to determine how effective it is. Three separate datasets, each of which represents a different scenario or set of tasks inside the Cloud-Fog environment, are used to thoroughly test the suggested methodology. These assessments' findings reveal that the suggested approach constantly outperforms the alternatives, illuminating its higher capacity for job optimization and QoS improvement.

5.1 Future Directions

Given the work's future projections, task offloading could include machine learning models, which could improve Quality of Service (QoS) optimization in Fog-cloud-based systems. By anticipating the job placement order, such models may give increased scalability and efficiency in optimizing QoS parameters.
Furthermore, new opportunities for enhancing resource scheduling in Fog computing are presented by the integration of cutting-edge technologies like federated learning and blockchain. Blockchain technology offers a decentralized, tamper-resistant ledger for documenting transactions and resource allocations, which can improve security, transparency, and trust in scheduling choices. Federated learning makes it possible for remote Fog nodes to collaborate on model training while maintaining data privacy, which makes QoS management more effective and customized.
Future study should also focus on resolving the scalability issues related to resource scheduling in extensive Fog settings. To enable the expansion of Fog computing deployments, scalable scheduling methods that can handle the growing complexity and volume of devices and applications will be necessary. Furthermore, in order to achieve comprehensive resource management in Fog environments, it will be imperative to investigate novel approaches for multi-objective optimization that take into account aspects like energy efficiency, dependability, and scalability in addition to quality of service.

References

Aazam, M., Zeadally, S., & Harras, K. A. (2018). Offloading in fog computing for iot: Review, enabling technologies, and research opportunities. *Future Generation Computer Systems*, *87*, 278–289.

Abd Elaziz, M., Abualigah, L., & Attiya, I. (2021). Advanced optimization technique for scheduling iot tasks in cloud-fog computing environments. *Future Generation Computer Systems*, *124*, 142–154.

Abdel-Basset, M., Mohamed, R., Chakrabortty, R. K., & Ryan, M. J. (2021). Iega: an improved elitism-based genetic algorithm for task scheduling problem in fog computing. *International Journal of Intelligent Systems*, *36*(9), 4592–4631.

Abdelmoneem, R. M., Benslimane, A., & Shaaban, E. (2020). Mobility-aware task scheduling in cloud-fog iot-based healthcare architectures. *Computer Networks*, 107348.

Abouaomar, A., Cherkaoui, S., Kobbane, A., & Dambri, O. A. (2019). A resources representation for resource allocation in fog computing networks. In *2019 ieee global communications conference (globecom)* (pp. 1–6).

Abouaomar, A., Mlika, Z., Filali, A., Cherkaoui, S., & Kobbane, A. (2021). A deep reinforcement learning approach for service migration in mec-enabled vehicular networks. In *2021 ieee 46th conference on local computer networks (lcn)* (p. 273-280). doi: 10.1109/LCN52139.2021.9524882

Abualigah, L., & Diabat, A. (2021). A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments. *Cluster Computing*, *24*, 205–223.

Abu-Amssimir, N., & Al-Haj, A. (2023). A qos-aware resource management scheme over fog computing infrastructures in iot systems. *Multimedia Tools and Applications*, 1–20.

Aburukba, R., AliKarrar, M., Landolsi, T., & El Fakih, K. (2020). Scheduling internet of things requests to minimize latency in hybrid fog cloud

computing. *Future Generation Computer Systems*, *111*, 539-551.

Agbaje, M., Ohwo, O., Ayanwola, T., Olufunmilola, O., et al. (2022). A survey of game-theoretic approach for resource management in cloud computing. *Journal of Computer Networks and Communications*, *2022*.

Akintoye, S. B., & Bagula, A. (2019). Improving quality-of-service in cloud-fog computing through efficient resource allocation. *Sensors*, *19*(6), 1267.

Al Ahmad, M., Patra, S. S., & Barik, R. K. (2020). Energy-efficient resource scheduling in fog computing using sdn framework. , 567–578.

Alam, M., Shahid, M., & Mustajab, S. (2024). Security challenges for workflow allocation model in cloud computing environment: a comprehensive survey, framework, taxonomy, open issues, and future directions. *The Journal of Supercomputing*, 1–65.

Alam, T. (2020). Cloud computing and its role in the information technology. *IAIC Transactions on Sustainable Digital Innovation (ITSDI)*, *1*(2), 108–115.

Albarracín, C. L., Venkatesan, S., Torres, A. Y., Yánez-Moretta, P., & Vargas, J. C. J. (2023). Exploration on cloud computing techniques and its energy concern. *Mathematical Statistician and Engineering Applications*, *72*(1), 749–758.

Albert, P., & Nanjappan, M. (2021). Whoa: Hybrid based task scheduling in cloud computing environment. *Wireless Personal Communications*, *121*(3), 2327–2345.

Alizadeh, M. R., Khajehvand, V., Rahmani, A. M., & Akbari, E. (2020). Task scheduling approaches in fog computing: A systematic review. *International Journal of Communication Systems*, *33*(16), e4583.

Alsaidy, S. A., Abbood, A. D., & Sahib, M. A. (2022). Heuristic initialization of pso task scheduling algorithm in cloud computing. *Journal of King Saud University-Computer and Information Sciences*, *34*(6), 2370–2382.

Al-Tarawneh, M. A. (2022). Bi-objective optimization of application placement in fog computing environments. *Journal of Ambient Intelligence and Humanized Computing*, *13*(1), 445–468.

Alzaqebah, A., Al-Sayyed, R., & Masadeh, R. (2019). Task scheduling based on modified grey wolf optimizer in cloud computing environment. In *2019 2nd international conference on new trends in computing sciences (ictcs)* (pp. 1–6).

Anang, D. S. (2020). *Locating self-clearing faults with machine learning of transients* (Unpublished doctoral dissertation). Howard University.

Aslanpour, M. S., Gill, S. S., & Toosi, A. N. (2020). Performance evaluation met-

rics for cloud, fog and edge computing: A review, taxonomy, benchmarks and standards for future research. *Internet of Things*, 100273.

Ayoubi, M., Ramezanpour, M., & Khorsand, R. (2021). An autonomous iot service placement methodology in fog computing. *Software: Practice and Experience*, *51*(5), 1097–1120.

Baburao, D., Pavankumar, T., & Prabhu, C. (2023). Load balancing in the fog nodes using particle swarm optimization-based enhanced dynamic resource allocation method. *Applied Nanoscience*, *13*(2), 1045–1054.

Bai, X., Su, X., & Zhu, R. (2021). Data-driven robust stochastic optimization for power systems operations. In *Uncertainties in modern power systems* (pp. 93–143). Elsevier.

Bansal, M., & Malik, S. K. (2020). A multi-faceted optimization scheduling framework based on the particle swarm optimization algorithm in cloud computing. *Sustainable Computing: Informatics and Systems*, *28*, 100429.

Bansal, S., & Aggarwal, H. (2023). A hybrid particle whale optimization algorithm with application to workflow scheduling in cloud–fog environment. *Decision Analytics Journal*, *9*, 100361.

Baranwal, G., & Vidyarthi, D. P. (2021). Fons: a fog orchestrator node selection model to improve application placement in fog computing. *The Journal of Supercomputing*, *77*, 10562–10589.

Bell, C. (2024). Cloud computing. In *Micropython for the internet of things: A beginner's guide to programming with python on microcontrollers* (pp. 413–424). Springer.

Bharathi, R., Abirami, T., Dhanasekaran, S., Gupta, D., Khanna, A., Elhoseny, M., & Shankar, K. (2020). Energy efficient clustering with disease diagnosis model for iot based sustainable healthcare systems. *Sustainable Computing: Informatics and Systems*, *28*, 100453.

Bhatt, P. C. P., & Sehgal, N. K. (2024). Cloud product development process. In *Project management in cloud applications* (pp. 63–88). Springer.

Bitam, S., Zeadally, S., & Mellouk, A. (2018). Fog computing job scheduling optimization based on bees swarm. *Enterprise Information Systems*, *12*(4), 373–397.

Chang, V., Sidhu, J., Singh, S., & Sandhu, R. (2023). Sla-based multi-dimensional trust model for fog computing environments. *Journal of Grid Computing*, *21*(1), 4.

Chhabra, A., Sahana, S. K., Sani, N. S., Mohammadzadeh, A., & Omar, H. A. (2022). Energy-aware bag-of-tasks scheduling in the cloud computing

system using hybrid oppositional differential evolution-enabled whale optimization algorithm. *Energies*, *15*(13), 4571.

Chhabra, A., Singh, G., & Kahlon, K. S. (2020). Qos-aware energy-efficient task scheduling on hpc cloud infrastructures using swarm-intelligence meta-heuristics. *Comput. Mater. Contin*, *64*, 813–834.

Chhabra, A., Singh, G., & Kahlon, K. S. (2021). Multi-criteria hpc task scheduling on iaas cloud infrastructures using meta-heuristics. *Cluster Computing*, *24*, 885–918.

Costa, B., Bachiega Jr, J., de Carvalho, L. R., & Araujo, A. P. (2022). Orchestration in fog computing: A comprehensive survey. *ACM Computing Surveys (CSUR)*, *55*(2), 1–34.

Das, R., & Inuwa, M. M. (2023). A review on fog computing: Issues, characteristics, challenges, and potential applications. *Telematics and Informatics Reports*, *10*, 100049.

Duan, S., Lyu, F., Wu, H., Chen, W., Lu, H., Dong, Z., & Shen, X. (2022). Moto: Mobility-aware online task offloading with adaptive load balancing in small-cell mec. *IEEE Transactions on Mobile Computing*.

Dubey, K., Kumar, M., & Sharma, S. C. (2018). Modified heft algorithm for task scheduling in cloud environment. *Procedia Computer Science*, *125*, 725–732.

Dzung, P. Q., Tien, N. T., Tuyen, N. D., & Lee, H.-H. (2015). Selective harmonic elimination for cascaded multilevel inverters using grey wolf optimizer algorithm. In *2015 9th international conference on power electronics and ecce asia (icpe-ecce asia)* (pp. 2776–2781).

Etemadi, M., Ghobaei-Arani, M., & Shahidinejad, A. (2020). Resource provisioning for iot services in the fog computing environment: An autonomic approach. *Computer Communications*, *161*, 109-131.

Gao, N., Xu, C., Peng, X., Luo, H., Wu, W., & Xie, G. (2020). Energy-efficient scheduling optimization for parallel applications on heterogeneous distributed systems. *Journal of Circuits, Systems and Computers*, 2050203.

Ghobaei-Arani, M., & Shahidinejad, A. (2022). A cost-efficient iot service placement approach using whale optimization algorithm in fog computing environment. *Expert Systems with Applications*, *200*, 117012.

Ghobaei-Arani, M., Souri, A., & Rahmanian, A. A. (2020). Resource management approaches in fog computing: a comprehensive review. *Journal of Grid Computing*, *18*(1), 1–42.

Ghobaei-Arani, M., Souri, A., Safara, F., & Norouzi, M. (2020). An efficient task scheduling approach using moth-flame optimization algorithm for

cyber-physical system applications in fog computing. *Transactions on Emerging Telecommunications Technologies*, *31*(2), e3770.

Ghosh, A. M., & Grolinger, K. (2020). Edge-cloud computing for internet of things data analytics: Embedding intelligence in the edge with deep learning. *IEEE Transactions on Industrial Informatics*, *17*(3), 2191–2200.

Goel, G., & Tiwari, R. (2023). Resource scheduling techniques for optimal quality of service in fog computing environment: A review. *Wireless Personal Communications*, 1–24.

Goel, G., Tiwari, R., Anand, A., & Kumar, S. (2022). Workflow scheduling using optimization algorithm in fog computing. In *International conference on innovative computing and communications: Proceedings of icicc 2021, volume 2* (pp. 379–390).

Gorlatova, M., Inaltekin, H., & Chiang, M. (2020). Characterizing task completion latencies in multi-point multi-quality fog computing systems. *Computer Networks*, *181*, 107526.

Grover, J., & Garimella, R. M. (2019). Optimization in edge computing and small-cell networks. In *Edge computing* (pp. 17–31). Springer.

Gupta, H., Vahid Dastjerdi, A., Ghosh, S. K., & Buyya, R. (2017). ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. *Software: Practice and Experience*, *47*(9), 1275–1296.

Gupta, S., Iyer, S., Agarwal, G., Manoharan, P., Algarni, A. D., Aldehim, G., & Raahemifar, K. (2022). Efficient prioritization and processor selection schemes for heft algorithm: A makespan optimizer for task scheduling in cloud environment. *Electronics*, *11*(16), 2557.

Hameed, T., Jamil, B., & Ijaz, H. (2024). Efficient resource scheduling in fog: A multi-objective optimization approach. *Proceedings of the Pakistan Academy of Sciences: A. Physical and Computational Sciences*, *61*(1).

Harnal, S., Sharma, G., Seth, N., & Mishra, R. D. (2022). Load balancing in fog computing using qos. *Energy Conservation Solutions for Fog-Edge Computing Paradigms*, 147–172.

Hasan, M. K., Sundararajan, E., Islam, S., Ahmed, F. R. A., Babiker, N. B. M., Alzahrani, A. I., ... others (2024). A novel segmented random search based batch scheduling algorithm in fog computing. *Computers in Human Behavior*, 108269.

Hazra, A., Adhikari, M., Amgoth, T., & Srirama, S. N. (2021). A comprehensive survey on interoperability for iiot: Taxonomy, standards, and future directions. *ACM Computing Surveys (CSUR)*, *55*(1), 1–35.

Heidari, A. A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., & Chen, H. (2019). Harris hawks optimization: Algorithm and applications. *Future generation computer systems*, *97*, 849–872.

Hong, C.-H., & Varghese, B. (2018). Resource management in fog/edge computing: A survey. *arXiv preprint arXiv:1810.00305*.

Hussain, M. M., Azar, A. T., Ahmed, R., Umar Amin, S., Qureshi, B., Dinesh Reddy, V., ... Khan, Z. I. (2023). Song: a multi-objective evolutionary algorithm for delay and energy aware facility location in vehicular fog networks. *Sensors*, *23*(2), 667.

Hussein, M. K., & Mousa, M. H. (2020). Efficient task offloading for iot-based applications in fog computing using ant colony optimization. *IEEE Access*, *8*, 37191–37201.

Ijaz, S., Munir, E. U., Ahmad, S. G., Rafique, M. M., & Rana, O. F. (2021). Energy-makespan optimization of workflow scheduling in fog–cloud computing. *Computing*, *103*, 2033–2059.

Jamil, B., Shojafar, M., Ahmed, I., Ullah, A., Munir, K., & Ijaz, H. (2020). A job scheduling algorithm for delay and performance optimization in fog computing. *Concurrency and Computation: Practice and Experience*, *32*(7), e5581.

Jeong, B., Baek, S., Park, S., Jeon, J., & Jeong, Y.-S. (2023). Stable and efficient resource management using deep neural network on cloud computing. *Neurocomputing*, *521*, 99–112.

Kashyap, V., & Kumar, A. (2022). Load balancing techniques for fog computing environment: Comparison, taxonomy, open issues, and challenges. *Concurrency and Computation: Practice and Experience*, *34*(23), e7183.

Kaur, K., Garg, S., Kaddoum, G., Gagnon, F., & Jayakody, D. N. K. (2019). Enlob: Energy and load balancing-driven container placement strategy for data centers. In *2019 ieee globecom workshops (gc wkshps)* (pp. 1–6).

Kaur, M., & Aron, R. (2021). Focalb: Fog computing architecture of load balancing for scientific workflow applications. *Journal of Grid Computing*, *19*(4), 40.

Kaur, N., Mittal, A., Lilhore, U. K., Simaiya, S., Dalal, S., & Sharma, Y. K. (2024). An adaptive mobility-aware secure handover and scheduling protocol for earth observation (eo) communication using fog computing. *Earth Science Informatics*, 1–18.

Khan, E., Garg, D., Tiwari, R., & Upadhyay, S. (2018). Automated toll tax collection system using cloud database. In *2018 3rd international conference on internet of things: Smart innovation and usages (iot-siu)* (pp. 1–5).

Kishor, A., & Chakarbarty, C. (2021). Task offloading in fog computing for using smart ant colony optimization. *Wireless personal communications*, 1–22.

Krishnaveni, S., Sivamohan, S., Sridhar, S., & Prabakaran, S. (2021). Efficient feature selection and classification through ensemble method for network intrusion detection on cloud computing. *Cluster Computing*, *24*(3), 1761–1779.

Kumar, M. S., & Karri, G. R. (2023). Eeoa: cost and energy efficient task scheduling in a cloud-fog framework. *Sensors*, *23*(5), 2445.

Li, G., Liu, Y., Wu, J., Lin, D., & Zhao, S. (2019). Methods of resource scheduling based on optimized fuzzy clustering in fog computing. *Sensors*, *19*(9), 2122.

Luo, J., Yin, L., Hu, J., Wang, C., Liu, X., Fan, X., & Luo, H. (2019). Container-based fog computing architecture and energy-balancing scheduling algorithm for energy iot. *Future Generation Computer Systems*, *97*, 50–60.

Lv, W., Chen, J., Cheng, S., Qiu, X., & Li, D. (2024). Qos-driven resource allocation in fog radio access network: A vr service perspective. *Mathematical Biosciences and Engineering*, *21*(1), 1573–1589.

Lyu, F., Ren, J., Cheng, N., Yang, P., Li, M., Zhang, Y., & Shen, X. S. (2020). Lead: Large-scale edge cache deployment based on spatio-temporal wifi traffic statistics. *IEEE Transactions on Mobile Computing*, *20*(8), 2607–2623.

Madhura, R., Elizabeth, B. L., & Uthariaraj, V. R. (2021). An improved list-based task scheduling algorithm for fog computing environment. *Computing*, *103*(7), 1353–1389.

Mahmud, R., Srirama, S. N., Ramamohanarao, K., & Buyya, R. (2019). Quality of experience (qoe)-aware placement of applications in fog computing environments. *Journal of Parallel and Distributed Computing*, *132*, 190–203.

Mahmudova, S. (2024). Using a digital twin to save energy in cloud computing. *Interdisciplinary Science Studies*(5).

Malleswaran, S. K. A., & Kasireddi, B. (2019). An efficient task scheduling method in a cloud computing environment using firefly crow search algorithm (ff-csa).

Mani, S. K., & Meenakshisundaram, I. (2020). Improving quality-of-service in fog computing through efficient resource allocation. *Computational Intelligence*, *36*(4), 1527-1547.

Mattia, G. P., Pietrabissa, A., & Beraldi, R. (2023). A load balancing algo-

rithm for equalising latency across fog or edge computing nodes. *IEEE Transactions on Services Computing*.

Mehmood, M., Javaid, N., Akram, J., Abbasi, S. H., Rahman, A., & Saeed, F. (2019). Efficient resource distribution in cloud and fog computing. In *Advances in network-based information systems: The 21st international conference on network-based information systems (nbis-2018)* (pp. 209–221).

Memari, P., Mohammadi, S. S., Jolai, F., & Tavakkoli-Moghaddam, R. (2022). A latency-aware task scheduling algorithm for allocating virtual machines in a cost-effective and time-sensitive fog-cloud architecture. *The Journal of Supercomputing*, *78*(1), 93–122.

Moh, T. C. M., & Moh, T. (2018). Prioritized task scheduling in fog computing.

Mohammadi, M., Bahrani-Pour, F., Ebrahimi-Mood, S., & Farshi, M. (2024). Security-aware resource allocation in fog computing using a meta-heuristic algorithm.

Mohammadzadeh, A., Chhabra, A., Mirjalili, S., & Faraji, A. (2024). Use of whale optimization algorithm and its variants for cloud task scheduling: a review. *Handbook of Whale Optimization Algorithm*, 47–68.

Mohammed, C. M., & Zeebaree, S. R. (2021). Sufficient comparison among cloud computing services: Iaas, paas, and saas: A review. *International Journal of Science and Business*, *5*(2), 17–30.

Murtaza, F., Akhunzada, A., ul Islam, S., Boudjadar, J., & Buyya, R. (2020). Qos-aware service provisioning in fog computing. *Journal of Network and Computer Applications*, 102674.

Mutlag, A. A., Abd Ghani, M. K., Mohd, O., Abdulkareem, K. H., Mohammed, M. A., Alharbi, M., & Al-Araji, Z. J. (2023). A new fog computing resource management (frm) model based on hybrid load balancing and scheduling for critical healthcare applications. *Physical Communication*, *59*, 102109.

Naas, M. I., Lemarchand, L., Boukhobza, J., & Raipin, P. (2018). A graph partitioning-based heuristic for runtime iot data placement strategies in a fog infrastructure. In *Proceedings of the 33rd annual acm symposium on applied computing* (pp. 767–774).

Naas, M. I., Lemarchand, L., Raipin, P., & Boukhobza, J. (2021). Iot data replication and consistency management in fog computing. *Journal of Grid Computing*, *19*, 1–25.

Naha, R. K., Garg, S., Chan, A., & Battula, S. K. (2020). Deadline-based dynamic resource allocation and provisioning algorithms in fog-cloud

environment. *Future Generation Computer Systems*, *104*, 131–141.

Naranjo, P. G. V., Pooranian, Z., Shojafar, M., Conti, M., & Buyya, R. (2019). Focan: A fog-supported smart city network architecture for management of applications in the internet of everything environments. *Journal of Parallel and Distributed Computing*, *132*, 274–283.

Nazeri, M., Soltanaghaei, M., & Khorsand, R. (2024). A predictive energy-aware scheduling strategy for scientific workflows in fog computing. *Expert Systems with Applications*, *247*, 123192.

Nazir, S., Shafiq, S., Iqbal, Z., Zeeshan, M., Tariq, S., & Javaid, N. (2018). Cuckoo optimization algorithm based job scheduling using cloud and fog computing in smart grid. , 34–46.

Ni, L., Zhang, J., Jiang, C., Yan, C., & Yu, K. (2017). Resource allocation strategy in fog computing based on priced timed petri nets. *IEEE Internet of Things Journal*, *4*(5), 1216–1228.

Ogundoyin, S. O., & Kamil, I. A. (2021). Optimization techniques and applications in fog computing: An exhaustive survey. *Swarm and Evolutionary Computation*, *66*, 100937.

Ogundoyin, S. O., & Kamil, I. A. (2023). Optimal fog node selection based on hybrid particle swarm optimization and firefly algorithm in dynamic fog computing services. *Engineering Applications of Artificial Intelligence*, *121*, 105998.

Okwu, M. O., Tartibu, L. K., Okwu, M. O., & Tartibu, L. K. (2021a). Grey wolf optimizer. *Metaheuristic Optimization: Nature-Inspired Algorithms Swarm and Computational Intelligence, Theory and Applications*, 43–52.

Okwu, M. O., Tartibu, L. K., Okwu, M. O., & Tartibu, L. K. (2021b). Particle swarm optimisation. *Metaheuristic Optimization: Nature-Inspired Algorithms Swarm and Computational Intelligence, Theory and Applications*, 5–13.

Peng, D., Sun, L., Zhou, R., & Wang, Y. (2023). Study qos-aware fog computing for disease diagnosis and prognosis. *Mobile Networks and Applications*, *28*(2), 452–459.

Premalatha, B., & Prakasam, P. (2024). Optimal energy-efficient resource allocation and fault tolerance scheme for task offloading in iot-fog computing networks. *Computer Networks*, *238*, 110080.

production system. (july 2020). *Logs of real parallel workloads from production systems.* Retrieved from `http://www.cse.huji.ac.il/labs/parallel/workload/logs.html` (Accessed on 2023-03-04)

Qu, Z., Wang, Y., Sun, L., Peng, D., & Li, Z. (2020). Study qos optimization

and energy saving techniques in cloud, fog, edge, and iot. *Complexity*, *2020*, 1–16.

Qureshi, M. S., Qureshi, M. B., Fayaz, M., Mashwani, W. K., Belhaouari, S. B., Hassan, S., & Shah, A. (2020). A comparative analysis of resource allocation schemes for real-time services in high-performance computing systems. *International Journal of Distributed Sensor Networks*, *16*(8), 1550147720932750.

Rafique, H., Shah, M. A., Islam, S. U., Maqsood, T., Khan, S., & Maple, C. (2019). A novel bio-inspired hybrid algorithm (nbiha) for efficient resource management in fog computing. *IEEE Access*, *7*, 115760-115773.

Rahbari, D., & Nickray, M. (2019). Low-latency and energy-efficient scheduling in fog-based iot applications. *Turkish Journal of Electrical Engineering and Computer Sciences*, *27*(2), 1406-1427.

Ramzanpoor, Y., Hosseini Shirvani, M., & Golsorkhtabaramiri, M. (2022a). Multi-objective fault-tolerant optimization algorithm for deployment of iot applications on fog computing infrastructure. *Complex & Intelligent Systems*, *8*(1), 361–392.

Ramzanpoor, Y., Hosseini Shirvani, M., & Golsorkhtabaramiri, M. (2022b). Multi-objective fault-tolerant optimization algorithm for deployment of iot applications on fog computing infrastructure. *Complex & Intelligent Systems*, *8*(1), 361–392.

Rehman, A. U., Ahmad, Z., Jehangiri, A. I., Ala'Anzy, M. A., Othman, M., Umar, A. I., & Ahmad, J. (2020). Dynamic energy efficient resource allocation strategy for load balancing in fog environment. *IEEE Access*, *8*, 199829–199839.

Rehman, S., Javaid, N., Rasheed, S., Hassan, K., Zafar, F., & Naeem, M. (2018). Min-min scheduling algorithm for efficient resource distribution using cloud and fog in smart buildings. , 15–27.

Roy, S., Banerjee, S., Chowdhury, K., & Biswas, U. (2017). Development and analysis of a three phase cloudlet allocation algorithm. *Journal of King Saud University-Computer and Information Sciences*, *29*(4), 473-483.

Sabireen, H., & Neelanarayanan, V. (2021). A review on fog computing: Architecture, fog with iot, algorithms and research challenges. *Ict Express*, *7*(2), 162–176.

Safa'a, S. S., Alansari, I., Hamiaz, M. K., Ead, W., Tarabishi, R. A., & Khater, H. (2023). ifogrep: An intelligent consistent approach for replication and placement of iot based on fog computing. *Egyptian Informatics Journal*, *24*(2), 327–339.

Salimian, M., Ghobaei-Arani, M., & Shahidinejad, A. (2021). Toward an autonomic approach for internet of things service placement using gray wolf optimization in the fog computing environment. *Software: Practice and Experience*, *51*(8), 1745–1772.

Saraswat, M., & Tripathi, R. (2020). Cloud computing: Comparison and analysis of cloud service providers-aws, microsoft and google. In *2020 9th international conference system modeling and advancement in research trends (smart)* (pp. 281–285).

Shen, M., Ma, B., Zhu, L., Mijumbi, R., Du, X., & Hu, J. (2017). Cloud-based approximate constrained shortest distance queries over encrypted graphs with privacy protection. *IEEE Transactions on Information Forensics and Security*, *13*(4), 940–953.

Sing, R., Bhoi, S. K., Panigrahi, N., Sahoo, K. S., Jhanjhi, N., & AlZain, M. A. (2022). A whale optimization algorithm based resource allocation scheme for cloud-fog based iot applications. *Electronics*, *11*(19), 3207.

Singh, A. K., Swain, S. R., Saxena, D., & Lee, C.-N. (2023). A bio-inspired virtual machine placement toward sustainable cloud resource management. *IEEE Systems Journal*.

Singh, K. D., & Singh, P. D. (2024). Qos-enhanced load balancing strategies for metaverse-infused vr/ar in engineering education 5.0. *Computer Applications in Engineering Education*, e22722.

Singh, S. P., Kumar, R., & Sharma, A. (2020). Efficient content retrieval in fog zone using nano-caches. *Concurrency and Computation: Practice and Experience*, *32*(2), e5438.

Singh, S. P., Sharma, A., & Kumar, R. (2020). Design and exploration of load balancers for fog computing using fuzzy logic. *Simulation Modelling Practice and Theory*, *101*, 102017.

Solutions, C. F. C. (2015). Unleash the power of the internet of things. *Cisco Systems Inc*.

Songhorabadi, M., Rahimi, M., MoghadamFarid, A., & Kashani, M. H. (2023). Fog computing approaches in iot-enabled smart cities. *Journal of Network and Computer Applications*, *211*, 103557.

Srirama, S. N. (2024). A decade of research in fog computing: Relevance, challenges, and future directions. *Software: Practice and Experience*, *54*(1), 3–23.

Sun, Y., Lin, F., & Xu, H. (2018). Multi-objective optimization of resource scheduling in fog computing using an improved nsga-ii. *Wireless Personal Communications*, *102*(2), 1369–1385.

Tadakamalla, U., & Menascé, D. A. (2021). Autonomic resource management for fog computing. *IEEE Transactions on Cloud Computing*, *10*(4), 2334–2350.

Talaat, F. M. (2022). Effective prediction and resource allocation method (epram) in fog computing environment for smart healthcare system. *Multimedia Tools and Applications*, *81*(6), 8235–8258.

Talaat, F. M., Ali, H. A., Saraya, M. S., & Saleh, A. I. (2022). Effective scheduling algorithm for load balancing in fog environment using cnn and mpso. *Knowledge and Information Systems*, *64*(3), 773–797.

Talaat, F. M., Saraya, M. S., Saleh, A. I., Ali, H. A., & Ali, S. H. (2020). A load balancing and optimization strategy (lbos) using reinforcement learning in fog computing environment. *Journal of Ambient Intelligence and Humanized Computing*, *11*(11), 4951–4966.

Tiwari, R., Sille, R., Salankar, N., & Singh, P. (2022). Utilization and energy consumption optimization for cloud computing environment. In *Cyber security and digital forensics: Proceedings of iccsdf 2021* (pp. 609–619).

Toor, A., ul Islam, S., Sohail, N., Akhunzada, A., Boudjadar, J., Khattak, H. A., ... Rodrigues, J. J. (2019). Energy and performance aware fog computing: A case of dvfs and green renewable energy. *Future Generation Computer Systems*, *101*, 1112–1121.

Tuli, S., Mirhakimi, F., Pallewatta, S., Zawad, S., Casale, G., Javadi, B., ... Jennings, N. R. (2023). Ai augmented edge and fog computing: Trends and challenges. *Journal of Network and Computer Applications*, 103648.

Varshney, S., & Singh, S. (2018). A survey on resource scheduling algorithms in cloud computing. *International Journal of Applied Engineering Research*, *13*(9), 6839–6845.

Vatanparvar, K., & Al Faruque, M. A. (2018). Control-as-a-service in cyber-physical energy systems over fog computing. In *Fog computing in the internet of things* (pp. 123–144). Springer.

Vijayalakshmi, R., Vasudevan, V., Kadry, S., & Lakshmana Kumar, R. (2020). Optimization of makespan and resource utilization in the fog computing environment through task scheduling algorithm. *International Journal of Wavelets, Multiresolution and Information Processing*, *18*(01), 1941025.

Vila, S., Guirado, F., Lerida, J. L., & Cores, F. (2019). Energy-saving scheduling on iaas hpc cloud environments based on a multi-objective genetic algorithm. *The Journal of Supercomputing*, *75*(3), 1483–1495.

Wadhwa, H., & Aron, R. (2022). Resource utilization for iot oriented framework using zero hour policy. *Wireless Personal Communications*, *122*(3), 2285–

2308.

Wang, P., Dong, K., Liu, H., Wang, B., Wang, W., & Wang, L. (2019). Research on task allocation and resource scheduling method in cloud environment. In *International conference on intelligent and interactive systems and applications* (pp. 560–567).

Wang, S., Zhao, T., & Pang, S. (2020). Task scheduling algorithm based on improved firework algorithm in fog computing. *IEEE Access*, *8*, 32385-32394.

Wang, T., Liang, Y., Jia, W., Arif, M., Liu, A., & Xie, M. (2019). Coupling resource management based on fog computing in smart city systems. *Journal of Network and Computer Applications*, *135*, 11–19.

Wang, W., Du, X., Shan, D., Qin, R., & Wang, N. (2020). Cloud intrusion detection method based on stacked contractive auto-encoder and support vector machine. *IEEE transactions on cloud computing*, *10*(3), 1634–1646.

Waqar, A., Skrzypkowski, K., Almujibah, H., Zagórski, K., Khan, M. B., Zagórska, A., & Benjeddou, O. (2023). Success of implementing cloud computing for smart development in small construction projects. *Applied Sciences*, *13*(9), 5713.

Xiao, L., Xu, M., Chen, Y., & Chen, Y. (2019). Hybrid grey wolf optimization nonlinear model predictive control for aircraft engines. *Applied Sciences*, *9*(6), 1254.

Xiao, X., & Zhao, M. (2022). Routing optimization strategy of iot awareness layer based on improved csa. *Neural Computing and Applications*, 1-12.

Yadav, A. M., Tripathi, K. N., & Sharma, S. (2022). An enhanced multi-objective fireworks algorithm for task scheduling in fog computing environment. *Cluster Computing*, 1–16.

Yang, X.-S. (2009). Firefly algorithms for multimodal optimization. In *International symposium on stochastic algorithms* (pp. 169–178).

Yin, C., Fang, Q., Li, H., Peng, Y., Xu, X., & Tang, D. (2024). An optimized resource scheduling algorithm based on ga and aco algorithm in fog computing. *The Journal of Supercomputing*, *80*(3), 4248–4285.

Yin, L., Luo, J., & Luo, H. (2018). Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing. *IEEE Transactions on Industrial Informatics*, *14*(10), 4712–4721.

Zhao, D., Luo, L., Yu, H., Chang, V., Buyya, R., & Sun, G. (2021). Security-sla-guaranteed service function chain deployment in cloud-fog computing networks. *Cluster Computing*, *24*, 2479–2494.

Dissemination of Work

Journals :

1. **SCIE Published-**Goel, G., & Tiwari, R. (2023). Resource Scheduling Techniques for Optimal Quality of Service in Fog Computing Environment: A Review. Wireless Personal Communications Springer, 1-24.
2. **SCIE Published-**Goel, G., & Tiwari, R. (2024). "IGWOA: Improved Grey Wolf Optimization Algorithm for Resource Scheduling in Cloud-Fog environment for delay sensitive applications". Peer-to-Peer Networking and Applications-Springer .
3. **Scopus Published-**Goel, G., & Tiwari, S. & Tiwari, R. (2023). Load Balancer using Whale-Earthworm Optimization for Efficient Resource Scheduling in the IoT-Fog-Cloud Framework. IJRITCC.
4. **ESCI Published-**Goel, G., & Tiwari, R. (2022). Resource scheduling in Fog environment using optimization algorithms for 6g networks. International Journal of Software Science and Computational Intelligence (IJSSCI), 14(1), 1-24.
5. **DOAJ-**Goel, G., & Tiwari, R. (2022). Task management in IoT-Fog-Cloud environment employing static scheduling Techniques. ENP Engineering Science Journal, 2(1), 13-20.

Conferences :

1. **Scopus Published-**Goel, G., Tiwari, R.(2021). Analysis of Resource Scheduling algorithms for optimization in IoT-Fog-Cloud System.
2. **Scopus Published-**Goel, G., & Tiwari, R. (2022, May). Dynamic resource allocation in Fog computing environment. In International Conference on Advancements in Interdisciplinary Research (pp. 85-93). Cham: Springer Nature Switzerland.
3. **Scopus Published-**Goel, G., Tiwari, R.(2018, December). Data preservation by hash algorithm for matrix multiplication over venomous cloud. In 2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC) (pp. 210-214). IEEE.

4. **Springer-**Goel, G., Tiwari, R.(2022). Workflow scheduling using optimization algorithm in Fog computing. In International Conference on Innovative Computing and Communications: Proceedings of ICICC 2021, Volume 2 (pp. 379-390). Springer Singapore.

UPES
UNIVERSITY WITH A PURPOSE

PLAGIARISM CERTIFICATE

1. We Dr. Shomik Tiwari (Internal Guide), _____(Co Guide/ External Guide) certify that the Thesis titled

Efficient Resource Scheduling for optimal Quality-of-Service in Fog computing

submitted by Scholar Mr/ Ms Gaurav Goel having SAP ID 500072316 has been run through a Plagiarism Check Software and the Plagiarism Percentage is reported to be 9 %.

2. Plagiarism Report generated by the Plagiarism Software is attached.

Signature of the Internal Guide

Signature of External Guide/ Co Guide

Signature of the Scholar

Thesis

ORIGINALITY REPORT

9%	5%	8%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	link.springer.com Internet Source	<1%
2	www.mdpi.com Internet Source	<1%
3	dr.ddn.upes.ac.in:8080 Internet Source	<1%
4	www.researchgate.net Internet Source	<1%
5	Submitted to Higher Education Commission Pakistan Student Paper	<1%
6	www.journaltocs.ac.uk Internet Source	<1%
7	Ranumayee Sing, Sourav Kumar Bhoi, Niranjan Panigrahi, Kshira Sagar Sahoo, Nz Jhanjhi, Mohammed A. AlZain. "A Whale Optimization Algorithm Based Resource Allocation Scheme for Cloud-Fog Based IoT Applications", Electronics, 2022 Publication	<1%