National College of Ireland

Dynamic Scheduling in Edge-Cloud Computing Environments using metaheuristic techniques

MSc Research Project
Research in Computing CA2

Javed Abidali Shaikh
Student ID: 21171581

School of Computing
National College of Ireland

Supervisor: Sean Heeney

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Javed Abidali Shaikh
……..

Student ID: 21171581
……

Programme: MSc in Cloud Computing **Year:** 2022-23
…… ……..
.

Module: Research in Computing
……

Supervisor Sean Heeney
……

Submission Due Date: 14th August 2023 at 2:00pm
……

Project Title: Dynamic Scheduling in Edge-Cloud Computing Environments using Metaheuristic Techniques
……

Word Count: ……

Page Count: ……..

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Javed Abidali Shaikh
……

Date: 14th August 2023 ……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Dynamic Scheduling in Edge-Cloud Computing Environments using Metaheuristic Techniques

Javed Abidali Shaikh
21171581

Abstract

Cloud computing has been expanded by fog/edge computing to the network's edge, where data sources and devices are located. For use cases like the Internet of Things (IoT) and other time-sensitive tasks, it attempts to provide low-latency, real-time data analysis and processing. In the fields of distributed computing and cloud computing, ideas like scheduling jobs, makespan, and resource utilization are fundamental. They gain considerably greater significance when used in a fog computing environment because of the specific attributes of fog/edge computing devices. The capacity of metaheuristic techniques to tackle dynamic, complicated, and frequently NP-hard optimization problems makes them ideal for dynamic job scheduling in cloud systems. In this study, we propose to apply particle swarm optimization (PSO) and artificial bee colony optimizer (ABC) algorithms for the task scheduling problem. The key objective is to minimize the makespan and thereby maximizing the usage of resources. The proposed work was implemented on the CloudSim simulation framework configured to represent the edge cloud infrastructure and the scheduling algorithms were trained on two workload datasets for benchmarking. The study assesses the performance of these two metaheuristic algorithms along with other baseline approaches and offers insights into how well they can improve scheduling performance and cloud resource management through extensive experiments and analysis.

1 Introduction

A strategy for providing on-demand computational resources and services, such as processing speed and data storage, over the Internet is called cloud computing (Mouradian et al.; 2017). The main restriction is the end devices' connectivity to the cloud as these connections are made via the Internet, which makes them unsuitable for a variety of cloud-based applications, including those that depend on latency. Additionally, cloud-based apps are frequently spread and comprised of several parts. As a result, it is normal practice to deploy individual application components across various clouds. Given that this new class of IoT applications requires a low response latency, the emergent Edge-Cloud computing paradigm appears to be promising (Jiao et al.; 2013).

Task scheduling and resource allocation are vital activities in contemporary dynamic cloud-enabled systems. The process of scheduling tasks entails the allocation of these tasks to processors that are both available and efficient. Furthermore, the process of resource allocation includes the formulation of a policy that governs the distribution of resources across different

tasks, with the aim of optimizing resource use. Nevertheless, the task of scheduling in the edge computational framework poses significant challenges as a result of various reasons. The key factor contributing to the significant variations in compute server capacity, speed, response time, and energy consumption between local nodes at the edge and remote cloud nodes is heterogeneity. Additionally, it is worth noting that computers located in the cloud and edge layers may exhibit various characteristics. Furthermore, the mobility aspect of the Edge paradigm results in a constant fluctuation of bandwidth between the data source and computation nodes. This necessitates the continuous implementation of dynamic optimization strategies in order to effectively fulfil the requirements of various applications. The stochastic nature of the Edge-Cloud environment manifests in various aspects, such as the rate at which tasks arrive, the duration of labour, and the resource demands, hence exacerbating the challenges associated with scheduling. In order to optimize resource utilization, save costs, and improve the quality of service for applications, dynamic task scheduling becomes necessary in stochastic environments (Tuli et al.; 2020).

1.1 Motivation

Fog computing utilizes edge devices and nodes in close proximity to the data source, hence mitigating data transmission latency and enhancing response times. The implementation of dynamic task scheduling aims to optimize the allocation of jobs to fog nodes in close proximity, hence reducing communication delays and improving the real-time performance of applications. Therefore, the waiting time for jobs, the length of the waiting queue and the time taken to complete a job from its initiation (*makespan*) are significant metrics for evaluating the quality of services provided by cloud providers (Ben Alla et al.; 2018). Consequently, it is crucial to take these limits into account during the design of a scheduling algorithm.

1.2 Research Problem

The research problem involves the development of efficient and adaptable task-scheduling algorithms for fog scenarios. These algorithms aim to optimize many conflicting objectives, such as minimizing makespan, optimizing resource utilization, and enhancing energy efficiency. This subject pertains to the resolution of challenges related to job allocation, load balancing, and resource management inside a fog environment that is characterized by dynamic changes and limited resources. The issue of scalability and heterogeneity in fog environments is tackled through the development of metaheuristic algorithmic approaches (Tawfeek and Elhady; 2016; Zhou et al.; 2018) that possess the capability to manage a significant quantity of jobs and resources, all the while ensuring computational efficiency. The proposed methods aim to improve the operational effectiveness, promptness, and optimal use of resources in fog computing systems. This, in turn, facilitates the effective implementation of time-critical applications in diverse fields.

1.3 Research Question

Q: Can metaheuristic approaches like PSO and ABC effectively address the task scheduling challenges in fog environments by optimizing the makespan time to improve resource utilization?

1.4 Research Objective

The main objective of this study is to enhance task scheduling and allocation in real-time or near-real-time situations in fog computing environments. This is achieved by taking into account the dynamic nature of these environments, the diverse characteristics of tasks, and the availability of resources. The proposed approaches, namely Particle Swarm Optimization (PSO) and Artificial Bee Colony (ABC) metaheuristic algorithms, are utilized to minimize the makespan and enhance resource allocation.

1.5 Research Contribution

The objective of this study is to enhance the allocation of tasks across fog nodes and cloud resources in real-time, while taking into account dynamic changes in task arrivals, execution times, and available resources. This optimization is conducted within the CloudSim simulation framework.

1) This study makes a significant contribution by employing Particle Swarm Optimization (PSO) and Artificial Bee Colony (ABC) techniques to address the work scheduling challenges in stochastic and heterogeneous fog computing environments. This entails modifying the exploration and exploitation processes of algorithms in order to successfully manage varying job demands and resource availability.

2) This study aims to compare the PSO and ABC techniques with baseline scheduling algorithms, namely Shortest Job First (SJF) and First-Come-First-Served (FCFS), in terms of optimizing the makespan for both space-shared and time-shared scheduler implementations.

1.6 Thesis structure

The present research report is structured into five distinct sections:

- Section 1 provides an introduction to the study background and outlines the factors that inspired the decision to pursue additional investigation in this area. Additionally, it elucidates the research topic and provides a response to the research question.
- Section 2 provides a comprehensive analysis of prior research conducted in the field, organizing it into subcategories according to its breadth, methodology, and relevance to the proposed strategy.
- Section 3 provides a comprehensive exposition of the research methodology in order to enhance the understanding of the research topic.
- Section 4 of this paper presents the design specifications of the suggested technique, accompanied by mathematical modeling and a theoretical foundation.
- Section 5 elucidates the implementation particulars of the proposed methodologies, expounding upon the specific tools and data sources employed.
- Section 6 provides a thorough examination of the study's outcomes and primary discoveries, along with an exploration of the implications derived from these findings.

- Section 7 provides a comprehensive summary of the research conducted and also highlighting the potential avenues for future research scope.

2. Related Work

2.1 Task scheduling in cloud

In order to optimize task scheduling performance and reduce non-optimal task allocation in cloud computing environments, (Zhang and Zhou; 2017) presented a two-stage strategy-based approach. In the initial phase, a job classifier that is inspired by the design principle of a Bayes classifier was developed. The utilization of past scheduling data is employed to categorize activities deployed in a cloud computing environment. The setting up of VMs was carried out in a manner to reduce the time required for the creation of VMs. (Hammoud et al.; 2019) proposed an academic approach that utilizes genetic algorithms and evolutionary game theory to investigate the challenge of establishing federated clouds with high profitability, while accounting for dynamic strategies. The researchers address the issue of optimizing the formation of federations to maximize overall profitability through the utilization of Genetic Algorithms (GA), while also ensuring stability within the federations. The study revealed that the utilization of the evolutionary game model yielded superior outcomes in terms of profitability and quality of service (QoS) as a result of its inherent ability to attain a state of stability.

In their study, (Luo et al.; 2018) proposed an improved iteration of the Particle Swarm Optimization (PSO) algorithm with the aim of minimizing the makespan and enhancing resource utilization within cloud computing systems. The authors suggest modifying the particle weights to account for variations in the number of repetitions and introducing random weights during the final stages of the Particle Swarm Optimization (PSO) approach. The primary goal is to mitigate the generation of local optimum solutions in the concluding PSO stages. Tawfeek and Elhady. (2016) presented the Ant Colony Optimization (ACO) algorithm to determine the best resource distribution for tasks in a dynamic cloud-based architecture to reduce the total system makespan. The scheduling approach was implemented through simulation using the Cloudsim toolkit package. The experimental findings, when compared to the First Come First Served (FCFS) and Round Robin (RR) algorithms, demonstrate that the ACO algorithm successfully meets the expected outcomes. The study proposed by (Elhady and Tawfeek; 2015) examined three potential methodologies suggested for the purpose of dynamic task scheduling in the context of cloud computing. The three methodologies belong to the domain of swarm intelligence was employed to address challenging or infeasible combinatorial problems. These methodologies drew inspiration from the behavior exhibited by ant colonies, particle swarms, and honeybees during foraging activities. The primary objective was to conduct an evaluation and comparative analysis of the various ways employed to minimize the makespan of a particular collection of jobs. The effectiveness of the algorithms is simulated using the toolkit package of CloudSim.

The researchers proposed in (Reddy and Phanikumar; 2018), an enhanced version of the ACO optimization algorithm with the objective of improving performance within the CloudSim

framework. This was accomplished by selecting several processors, minimizing the makespan, and achieving a high convergence speed in the shortest possible time. The primary objective of modified ACO (MACO) is the intentional allocation of pheromones to virtual machines, taking into consideration their respective efficiency. Additionally, MACO also considers factors such as processing speed, makespan, and bandwidth when allocating jobs.

2.2 Task scheduling in Fog/Edge Networks

In a First-Come-First-Serve (FCFS) scheduling system proposed by (Mathew et al.; 2014), a new task upon arrival was appended to the tail of the queue and the initial task in the queue was always executed first. This approach was characterized by its straightforward implementation. The round-robin (RR) scheduling approach is derived from FCFS method, which aims to allocate resources to tasks at predetermined time intervals. One of the benefits of employing this particular method is the implementation of load-balancing. The priority scheduling algorithm presented by (Wu et al.; 2013) categorizes jobs according to their priority and the consideration of these priorities based on QoS factors where the tasks were given resources that have the most efficient completion time. The scheduling algorithm known as Multi-objective Heterogeneous Earliest Finish Time (MOHEFT) proposed by (Durillo and Prodan; 2014) was based upon the concept of Pareto solutions. The optimization of makespan and cost is rooted in the utilization of workflow apps within the context of the Amazon commercial cloud. The multi-objective algorithm MOHEFT is highly appealing due to its inherent versatility. Empirical evidence has demonstrated that in certain instances, a modest 5% improvement in makespan can result in a noteworthy reduction of expenses by half.

The Min-Min scheduling algorithm operates by selecting the smallest job from the pool of available tasks and assigning it to a resource. The selected job is then executed for the lowest amount of time required to complete it. This approach results in an increase in the makespan. The Max-Min approach is a strategy that involves selecting the task with the longest duration from a given set of tasks and assigning it to the machines with the highest processing speed for execution. This approach necessitates that smaller jobs endure longer waiting periods, resulting in an augmented makespan. Nevertheless, this approach demonstrates a superior makespan in comparison to the methods employed by other researchers (Gasmi et al.; 2022). The task scheduling in the iFogsim environment utilizes the FCFS, concurrent, and DP scheduling approaches, as discussed in (Bittencourt et al.; 2017). The researchers presented two case studies and conducted an analysis of the findings, focusing on variables such as delay, total network utilization, and the number of application modules. These variables were examined in relation to the number of users. The findings indicate that the delay-priority technique yields the lowest latency, while the concurrent method results in the least network use.

In their study, (Wang et al.; 2017) employed ACO as a methodology for addressing the resource requirements of mobile cloud computing. The aforementioned approach involves the execution of offloaded tasks on fog devices (FDs) with the aim of achieving objectives related to delay, completion time, and energy consumption. The temporal sequencing of their simulation is contingent upon the quantity of cycles, tasks, and ants. Wang et al. (2014) employed a multi-objective genetic algorithm (GA) to effectively minimize energy consumption and enhance profitability for the service provider. The Pareto principle was employed to determine the most

optimal selection among the available options, taking into consideration the prevailing needs at a given time. The simulation findings obtained from CloudSim demonstrate a reduction in energy consumption rates for the service provider by 44.46%.

Yassa et al. (2013) introduced a methodology that combines PSO with HEFT. The primary objective of the algorithm is to maximize efficiency in terms of minimizing makespan, cost, and power usage. The algorithm commences by initializing the position and velocity of particles in the PSO technique. The HEFT algorithm is iteratively employed multiple times in order to identify an optimal solution that minimizes the makespan. The findings indicate that their methodology not only exhibits superior cost and power efficiency, but also enhances the makespan. The scheduling challenge in the fog computing environment was addressed by the authors (Bitam et al.; 2017) through the proposal of a bio-inspired solution that utilized the Bees Life algorithm. The proposed technique was predicated on the allocation of a collection of jobs among all functional dependencies (FDs). The examination of the CPU's execution time and allocated memory via FDs subsequent to simulation revealed that this approach exhibits superior performance compared to both PSO and GA.

Zade et al. (2021) proposed a multi-objective approach that combines the Hitchcock bird algorithm and fuzzy signature to address the job scheduling problem in cloud computing. The authors saw enhancements in makespan and resource utilization when comparing their proposed approach to both the Moth Search Algorithm with Enhanced Multi-verse optimizer and the Fuzzy Modified PSO. The utilization of Opposition-based learning was also employed in the study (Agarwal and Srivastava.; 2021) to optimize PSO for the purpose of job scheduling within a cloud computing environment. The researchers enhanced the convergence of the conventional PSO algorithm, as well as addressed the issues of energy consumption and makespan.

2.3 Dynamic Task Scheduling Approaches

The primary aim of the study conducted by Dong et al. (2020) is to minimize expenses, with a specific focus on addressing intricate scheduling challenges associated with several jobs. Given the aforementioned circumstances, the authors proposed a technique rooted in a deep reinforcement learning (DRL) framework. This approach aims to dynamically allocate tasks with interdependencies to cloud servers, hence minimizing the overall execution time of activities. Furthermore, to effectively tackle the challenge posed by the substantial dimensionality and intricacy, the researchers employ a Deep Reinforcement Learning (DRL) approach referred to as Deep Q-Network (DQN). Liu et al. (2017) put forth a proposition for a two-phase framework that effectively tackles the challenges of resource allocation for virtual machines (VMs) on servers and power consumption management on individual servers. The authors commence their study by employing Deep Reinforcement Learning (DRL) as a means to accomplish VM resource allocation on servers, constituting the preliminary phase of the project. During the second phase, the Long Short-Term Memory (LSTM) model and shared weights were employed to efficiently regulate the local power consumption of the servers.

Huang et al. (2022) proposed a PSO technique that relies on heuristics and is grounded in a Lyapunov framework. The objective of this technique is to optimize energy usage in the context of resource allocation inside a fog environment. The optimization of energy consumption in

IoT systems involves considering the energy utilized by IoT nodes, transmission power utilization, and energy consumption while processing at the fog node. This balance enables the scheduling of jobs with minimal energy consumption. In their study, Li et al. (2019b) employed a hybrid genetic algorithm-particle swarm optimization (GA-PSO) approach to develop a load balancing mechanism specifically tailored for molecular dynamics simulations conducted on heterogeneous supercomputing systems. The results of their study exhibited a notable enhancement in the efficacy of parallel computing. Long et al. (2022) presented a novel approach for addressing the challenge of integrating new jobs into flexible job-shops. Specifically, the proposed solution involves the utilization of a dynamic self-learning artificial bee colony optimization algorithm, which aims to effectively solve the dynamic flexible job-shop scheduling problem (DFJSP). By strategically organizing the processing sequence of jobs and establishing appropriate relationships between operations and machines, it is possible to reduce the makespan, enhance the economic benefits of the job-shop, and raise the utilization rate of processing equipment.

The literature review conducted on task scheduling in Fog networks highlights the dynamic and ever-changing characteristics of Fog computing systems. The paper provides a comprehensive perspective on the difficulties and potential advantages related to enhancing job allocation in such settings, establishing a basis for future scholarly investigation and advancement in this domain. The selection of PSO and ABC metaheuristic techniques for real-time job scheduling, with the objective of minimizing the makespan and enhancing resource usage, was informed by prior research.

Table-I: Comparison of previous literature on task scheduling based on metaheuristic and deep learning approaches

Research	Technique	Merits	Shortcomings	Test Environment
Mandal and Acharyya. (2015)	Meta-heuristic swarm optimization	Low makespan	Low reliability	GCC Compiler
Ramezani et al. (2013)	MOPSO Improved PSO	Low execution time	High energy consumption	CloudSim
He et al. (2016)	PSO	Low makespan, low energy consumption	Low scalability	CloudSim
Madni et al. (2019)	CSA Cuckoo Search	Low makespan, high resource utilization	Low reliability	CloudSim
Raju et al. (2013)	Hybrid ACO, CSA	Low makespan, High throughput	Low resource utilization	Real test environment
Sharma and Garg (2019)	Hybrid GA with HS	Low makespan, low energy consumption	Low reliability	CloudSim
Gabi et al. (2019)	CSO & PSO	Low makespan, high resource utilization	High computational cost	CloudSim
Chaudhary et al. (2017)	NPSO	Low monetary cost	Low reliability	CloudSim
Yuan and Bi. (2019)	Hybrid PSO with GA	High throughput, low energy consumption	High makespan	Real Test environment
N.R. Rajalakshmi et al. (2019)	Deep Reinforcement Learning (DRL)	VM consolidation for energy efficiency	Total hosts restricted	Workload Dataset
El-Boghdadi et al. (2019)	Deep Reinforcement Learning (DRL)	Offline cloud resource scheduling	No complex scheduling problems	Workload Dataset

3. Research Methodology

3.1 Scheduling process in Fog/Edge Clouds

Resources are allocated not solely according to standard procedures, but rather based on the existing workload, observable applications, and requests for virtual machines (VMs). The fundamental objective of cloud service providers is to leverage the cloud infrastructure in order to minimize execution time and costs, while also optimizing resource utilization. A cloud scheduler employs several ways to enhance resource utilization and minimize both processing expenses and time requirements during the allocation of jobs to virtual machines (VMs). The concept of fog extends cloud services to the edge of the network. The allocation of application execution in fog computing is contingent upon the utilization of an efficient task-scheduling mechanism.

Figure 1 illustrates the hierarchical structure of the network, showcasing the location of various components such as devices, fog layer, cloud layer, and scheduler. Customers at the device layer have the ability to utilize fog/cloud resources in order to access and utilize various services. This is made possible due to the presence of storage, compute, and networking

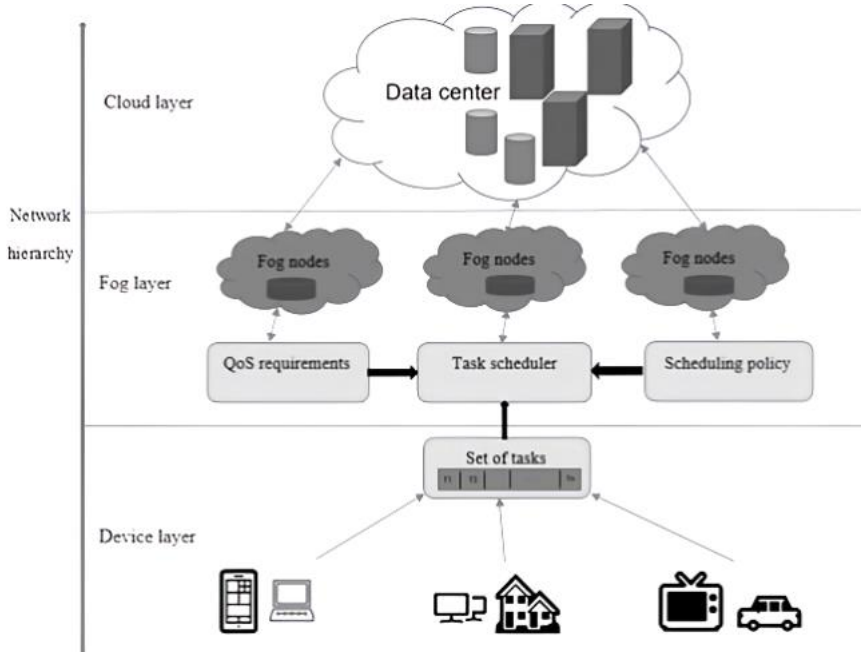

Figure 1: Scheduling process in a fog/edge cloud network

capabilities within these devices. In addition, a scheduler has the ability to guarantee static resource allocation for new requests, as well as the option to optimize and realign both static and dynamic resource allocation (Rahimikhanghah et al.; 2022).

In fog computing, task scheduling refers to allocating the proper resources to a task. The utilization of appropriate resources in this context serves to reduce the overall completion time of tasks (makespan), enhance the quality of service (QoS) provided to consumers, and optimize efficiency. Virtualization plays a crucial role in the realm of cloud computing as it facilitates the sharing of virtual resources and services that are distinct from the physical hardware on which they are built. Alternatively, the application of dynamic allocation of virtual machines (VMs) to servers can be employed as a means to decrease server demand and mitigate the energy consumption of data centers. In a cloud computing context, scheduling can be categorized into two distinct tiers. Initially, it is advisable to allocate virtual machines (VMs) to hosts that are currently unoccupied. By implementing a methodical scheduling approach, it becomes possible to achieve equilibrium in terms of system load distribution and energy utilization. The allocation of virtual machines (VMs) onto physical computers can be categorized as either dynamic or static, and several methods of VM migration can be employed. The scheduling of arrival tasks should be done in such a way that they are allocated to a virtual machine (VM) with adequate resources to meet the task requirements. Certain scheduling approaches are designed to deactivate idle virtual machines (VMs) in order to optimize resource use, conserve energy, and thereby improve resource utilization.

3.2 Proposed Scheduler architecture

Figure 2 depicts the system architecture for task scheduling based on the Particle Swarm Optimization (PSO) algorithm. Customers who are interested in using cloud computing services shall approach the cloud provider with their service requests. Subsequently, the cloud

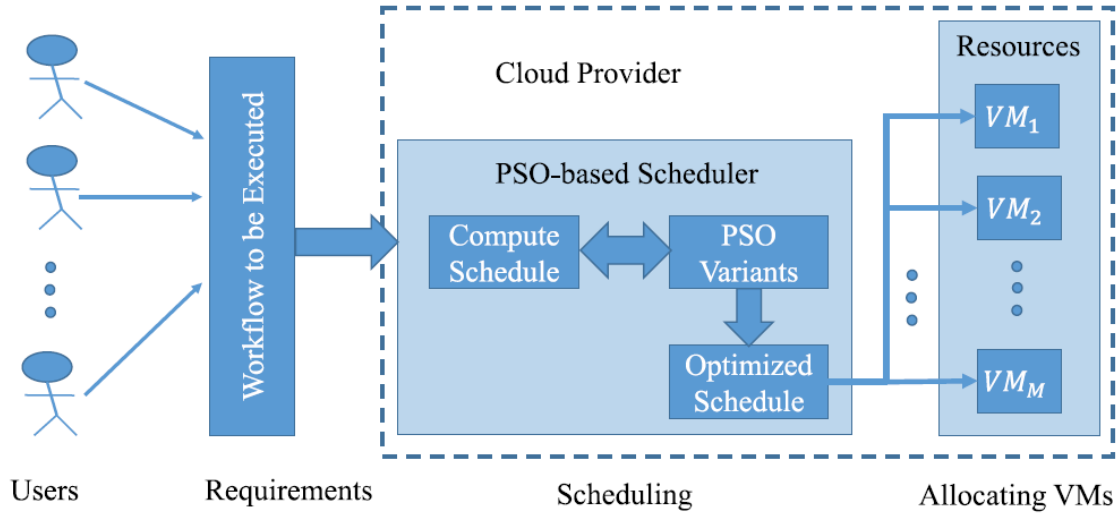

Figure 2: PSO-Scheduler architecture

provider is required to identify an efficient scheduling solution for the workflow application request that has been provided. The proposed scheduling solution represents the most efficient task execution nodes (VMs) to minimize the overall makespan of the system. The same scheduling architecture is applicable to artificial bee colony scheduler also.

4. Design Specifications

4.1 Problem Formulation

In every swarm intelligence algorithm that operates on a population-based approach, the optimization process is comprised of two distinct phases: global search and local search. Achieving a suitable equilibrium between the two stages enables the algorithm to effectively converge towards the optimal global position within a reasonable timeframe. In order to optimize the search method, it is recommended that algorithms prioritize the utilization of the global search operator over the local search operator during the initial stages. This enables the algorithm to effectively traverse a wider range of regions inside the search space, necessitating a meticulous search approach in order to attain the global optimum (Huang et al.; 2020). In this work, obtaining the global optimum involves the makespan minimization based on certain real-time constraints. A fitness function is defined that asserts the quality of the swarm optimization algorithm's solution. The optimization problem can be phrased as an aim to minimize the makespan in task scheduling. The term "makespan" refers to the duration required for all tasks to finish their execution. The objective is to identify a task-to-resource assignment i.e cloudlet to VMs that minimizes this duration. The following is the standard formula for minimizing the makespan in a task scheduling problem:

$$M = \max_{i=1}^{n} \max_{j=1}^{m} C_{ij}$$

where,

n - number of tasks

m - number of resources like fog nodes or edge devices

C_{ij} - the completion time of task i on resource j

p_{ij} - the processing time of task i on resource j

This research presents the implementation of two popular meta-heuristic algorithms, PSO and ABC for achieving the objective of minimizing the makespan. The next sub-section discusses in details the algorithm design and its mathematical modeling in sufficient detail. Finally, CloudSim, the simulation environment used in this research for modeling the fog networks is introduced.

4.1 PSO-based Task Scheduler

The evaluation of a particle's performance is quantified by a fitness value, which may be determined by defining a fitness function that measures the effectiveness of the particle's position.

$$f = \text{Min}\left(\frac{V_{time}}{VR_{utilization}}\right)$$

The variable V_{time} is used to represent the execution time of virtual machines (VMs) for the completion of all tasks. On the other hand, $VR_{utilization}$ is employed to indicate the resource utilization of VMs while the tasks are being executed. Particles in the search process maintain self-updating capabilities through the monitoring and recording of their two most optimal places. The most widely recognized position, referred to as the local best position, is the individual position that has achieved the highest fitness value thus far for the particle itself. One well recognized position, referred to as the global best position, is the optimal location within the entire population. The pseudo-code for the implementation is presented in Algorithm-1.

Algorithm-1: PSO Task Scheduler pseudocode.

Input: Task, Particles

Output: gBest

for each P_i do

$pBest =$ Generate_initial_position (P_i)

foreach p Best of particle P_i do

gBest = Max (pBest t_1, pBest t_2, ….)

repeat

j←1

while $j \leq m$ **do**

Select the task t_j

Calculate_est (t_j)

Allocate_task (t_j)

j++

foreach particle P_i **do**

Calculate cur_particle_fit: (current particle)

if cur_particle_fit ^{i} pBest i_fit **then**

Update $(pBest_i)$

if cur_particle_fit $<$ g Best $_i$_fit **then**

Update (gBest $_i$)

if the termination condition is met **then**

Output $4g$ Best $_i$; **Break**

else

foreach particle P_i **do**

Update($P_{i\_velocity)}$;

Update(P_i position);

until the termination condition is met.

4.2 ABC-based Task Scheduler

The method operates on a comparable notion of workload balancing for virtual machines (VMs). The ABC method is utilized to assess the workload of a virtual machine and then determine whether it is experiencing overload, is underutilized, or is operating at an optimal level of balance. The task's high priority is not aligned with the overload virtual machine, resulting in jobs being queued for the lightweight virtual machine. The subsequent stage involves referring to these assignments as scout bees. The Load Balancing technique inspired by ABC has been shown to effectively decrease the response time of VMs and minimize the waiting time of tasks. The probability of calculating a new solution surrounding a food source is calculated by onlooker bees using the following equation:

$$p_i(t) = \frac{f_i(t)}{\sum_{i=1}^{TS} fit_i(t)}$$

$fit_i(t)$ – fitness value of task source i

TS – Total number of task sources

The pseudo-code for the implementation is presented in Algorithm-1.

Algorithm-2: ABC Task Scheduler pseudocode.

Require: Cloudlet $_{\text{list}}$, VM $_{\text{list}}$, Datacenter list, fac $_{LB}$

Groups $(q) \leftarrow$ divide (Cloudlet $_{\text{list}}$)

for i = 1 to q **do**

length $_i \leftarrow$ lengthofgroupk (Groups $_i$)

end for

for k = 1 to q **do**

Cloudlet $_L \leftarrow$ max (Groups $_k$)

while Group $_k \geq$ Groups $_i \mid i = 1$ ... qand $i \neq k$ **do**

for s = 1 to n **do**

Datacenter $_s \leftarrow$ select (Datacenter list)

if $fac_{LB} \leq VM_S$ Assigned (Datacenter) **then**

assign ($Cloudlet_L$, Datacenter $_{i \neq s}$(VM $_{\text{leastLoad}}$))

else

Decrement ($length_k$)

end if

end for

end while

end for

4.3 CloudSim Architecture

CloudSim, an open-source platform, is employed for the purpose of simulating cloud computing services and infrastructure. The software application is developed by the CLOUDS Labs and is implemented entirely using the Java programming language. Simulating and modeling a cloud computing environment is commonly employed to duplicate tests and outcomes, serving as a means to evaluate a hypothesis prior to software development.The CloudSim toolkit facilitates the modeling of both system and behavior aspects of Cloud system components, including data centers, virtual machines (VMs), and resource provisioning policies. The system has generic methods for provisioning applications that can be easily expanded and require minimal effort. At present, the software facilitates the representation and emulation of Cloud computing environments that encompass both individual clouds and interconnected clouds.

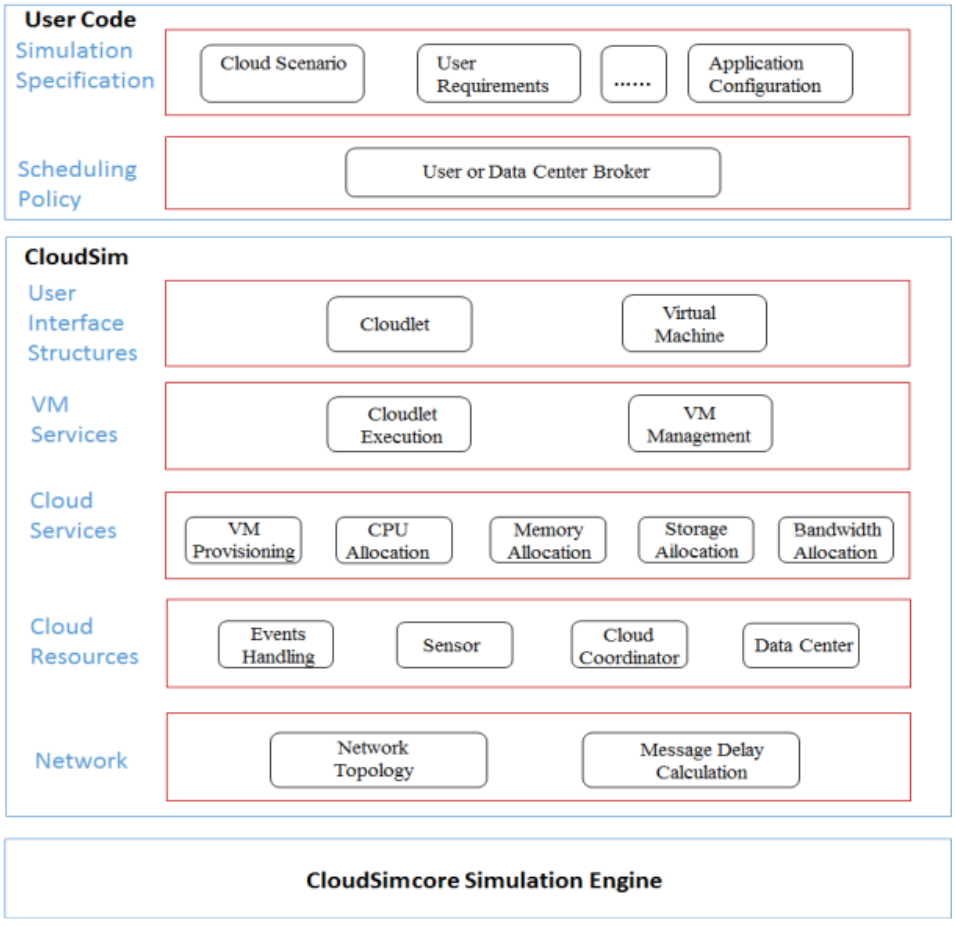

Figure 3: CloudSim architecture

Figure 3 depicts the intricate design of the CloudSim software framework, showcasing its various architectural components. The *CloudSim simulation layer* offers assistance in the modeling and simulation of virtualized Cloud-based data center infrastructures. This includes specific management interfaces for virtual machines (VMs), memory, storage, and bandwidth. This layer is responsible for addressing fundamental concerns, including the allocation of hosts to virtual machines. In order to investigate the effectiveness of various policies in allocating hosts to virtual machines (VMs), a Cloud provider must execute their techniques at the VM

provisioning layer, where the proposed PSO and ABC in addition to other baselines algorithms discussed earlier were written in this work.

The *datacenter* serves as the fundamental hardware component of a cloud system, and its modeling is of utmost importance. This course provides methodologies for determining the allocation rules for virtual machines (VMs), as well as addressing the functional requirements of the datacenter and other relevant considerations.

The *cloudlet* class represents any task that is performed on a VM, including processing tasks, memory access tasks, file update tasks, and so on. The class possesses methods that exhibit similarity to those found in the VM class. It also retains parameters that describe many aspects of a job, including its length, size, execution time, status, cost, and history.

A datacenter broker is a service that acts as a representative for the user or consumer. The responsibility of managing the functioning of VMs, encompassing their creation, management, termination, and the allocation of cloudlets to them, lies within its purview.

The implementation of the proposed scheduling policies on CloudSim and the execution of those under real-time workload datasets on the cloud with the set parameters will be the discussion of the next section.

5. Implementation

The whole code implementation of the scheduling algorithms, the datacenter creation, creating the edge simulation environment are all done at the structured directories of the CloudSim simulation engine. Task scheduling in CloudSim involves setting up a simulation environment to model and evaluate different task scheduling algorithms in a cloud computing context.

A. DEFINE CLOUDSIM ENTITIES:

Create classes to represent different CloudSim entities, such as Datacenter, Host, VM (Virtual Machine), and Cloudlet (Task). In this work, *Simulation.java* code is used to create the simulation environment with cloudlet scheduler type, broker type, number of cloudlets, number of VMs etc., (see Figure 4).

```
public Simulation(int cloudletSchedulerType, int numOfCloudlets, int numOfVMs, int brokerType, int fitnessType,
    this.cloudletSchedulerType = cloudletSchedulerType;
    this.numOfCloudlets = numOfCloudlets;
    this.numOfVMs = numOfVMs;
    this.brokerType = brokerType;
    this.fitnessType = fitnessType;
    this.rng = rng;
    this.silent = silent;
```

Figure 4: Simulation.java Class

B. *SET UP CLOUD RESOURCES*:

- Create Datacenter entities representing cloud data centers.
- Define Hosts with their processing capacities, storage, and bandwidth.
- Create VMs with specific characteristics (e.g., MIPS, RAM, storage).

C. GENERATE CLOUDLETS (TASKS):

- Create Cloudlet objects to represent the tasks that need to be scheduled.

- Specify Cloudlet characteristics like length (execution time), input and output file sizes, and utilization of CPU and bandwidth.

D. IMPLEMENT TASK SCHEDULING ALGORITHM:

- Choose the scheduling algorithm (*test.java*), that initiates the running of the proposed algorithms, ABC and PSO. In addition, the proposed methods compared with the baseline algorithms included with CloudSim, namely FCFS and SJF.
- Implement the scheduling logic in Java code, considering task-to-VM allocation based on the chosen algorithm.

E. RUN THE SIMULATION:

- Configure simulation parameters such as simulation duration and scheduling interval.
- Instantiate CloudSim objects, including the CloudSim class itself, Broker, and DatacenterBroker.
- Add VMs and Cloudlets to the broker.
- Two workload files were considered in this simulation run: High-Performance Computing Center North (HPC2N) workload log consisting of simulations for running simulations, computational modeling, and other data-intensive tasks. NASA-iPSC-19923 workload is a historical benchmark dataset that includes information about parallel programs, their execution times, and various performance metrics.
- Start the simulation using CloudSim's **startSimulation()** method.

F. COLLECT AND ANALYZE RESULTS:

- Monitor the simulation progress using CloudSim's simulation events and log outputs.
- Capture relevant performance metrics such as makespan, response time, etc.
- Analyze the simulation results to evaluate the effectiveness of your task scheduling algorithm. *calculateStatistics()* was used to compute the min, max, average and standard deviation values of *makespan* for all the algorithms simulated under different parameter settings. The VMs, VM speed, cloudlet assignments and broker operations were logged into a text file for further analysis.

6. Evaluation

The evaluation of the results for the proposed PSO and ABC scheduling algorithms along with the FCFS scheduler, SJF scheduler are considered for the NASA workload scenarios, different number of cloudlets with both low and high heterogeneity for both space-shared and time-shared schedulers. The results are populated below for each experimental case.

6.1 Case Study-1: On NASA workload for space-shared scheduling policy

Experiment-1: Scenario 0 - low number of cloudlets, high heterogeneity

Scheduler/MakeSpan Metrics	FCFS	SJF	PSO	ABC
Average	90.9607	89.0959	27.6844	26.9257
Minimum	55.9489	51.9007	23.1852	23.0019
Maximum	129.2697	133.4309	33.6020	32.7110
Std	24.7958	27.7556	3.0812	2.8419

Experiment-2: Scenario 1 - low number of cloudlets, low heterogeneity

Scheduler/MakeSpan Metrics	FCFS	SJF	PSO	ABC
Average	30.7524	29.9404	28.8337	28.2485
Minimum	29.5088	28.2969	27.7181	26.6902
Maximum	32.7854	31.2136	30.1401	29.3722
Std	1.1249	0.8109	0.8733	0.9255

Experiment-3: Scenario 2 - medium number of cloudlets, high heterogeneity

Scheduler/MakeSpan Metrics	FCFS	SJF	PSO	ABC
Average	892.4073	894.4111	369.5945	264.4073
Minimum	531.3036	519.1664	240.4661	231.1631
Maximum	1310.1034	1306.1075	551.2373	317.2269
Std	268.5950	272.6343	90.7772	28.4100

Experiment-4: Scenario 3 – medium number of cloudlets, low heterogeneity

Scheduler/MakeSpan Metrics	FCFS	SJF	PSO	ABC
Average	300.2538	299.1365	278.4629	276.7469
Minimum	290.9952	294.6626	269.4633	268.6151
Maximum	317.1105	304.3433	286.3509	282.5472
Std	7.2863	3.2480	5.2394	4.5820

6.2 Case Study-1: On NASA workload for time-shared scheduling policy

Experiment-1: Scenario 0 - low number of cloudlets, high heterogeneity

Scheduler/MakeSpan Metrics	FCFS	SJF	PSO	ABC
Average	90.8936	88.9932	27.5614	26.7912
Minimum	55.9340	51.7404	23.0434	22.7143
Maximum	129.2260	133.3391	33.5249	32.6801
Std	24.8114	27.7485	3.1092	2.9149

Experiment-2: Scenario 1 - low number of cloudlets, low heterogeneity

Scheduler/MakeSpan Metrics	FCFS	SJF	PSO	ABC
Average	30.6656	29.8221	28.7758	28.1126
Minimum	29.3949	28.1642	27.6744	26.5919
Maximum	32.8773	31.2141	30.1486	29.2759
Std	1.2054	0.8452	0.8761	0.9522

Experiment-3: Scenario 2 - medium number of cloudlets, high heterogeneity

Scheduler/MakeSpan Metrics	FCFS	SJF	PSO	ABC
Average	891.6590	893.4330	368.2089	261.8907
Minimum	530.4117	517.7579	238.7128	228.0479
Maximum	1309.5995	1305.2913	550.3147	315.4462
Std	268.6124	272.6698	91.1793	28.7227

Experiment-4: Scenario 3 - medium number of cloudlets, low heterogeneity

Scheduler/MakeSpan Metrics	FCFS	SJF	PSO	ABC
Average	298.6425	297.4734	276.7122	274.8653
Minimum	289.2978	292.9940	267.5950	266.7206
Maximum	315.5544	302.7690	284.3548	280.7276
Std	7.2698	3.3196	5.2271	4.5936

6.3 Discussions

The experiments were conducted on both the case studies on the NASA workload file for the two cloudlet scheduler policies provided by CloudSim, namely space-spared and time-shared. The *CloudletSchedulerSpaceShared* class implements a scheduling strategy for virtual machines to execute their Cloudlets. It is assumed that there will be only one Cloudlet assigned to each VM. Additional Cloudlets will be placed on a queue for future utilization. Additionally, it takes into account the fact that the transfer of Cloudlets to the Virtual Machine (VM) occurs prior to the execution of the Cloudlet. That is to say, while Cloudlets have a waiting period for CPU allocation, data transfer occurs promptly upon the submission of Cloudlets. The scheduler does not take into account the priorities of Cloudlets when determining the order of execution. When Cloudlets have stated priorities, the scheduler disregards them. The CloudletSchedulerTimeShared class defines a policy for executing Cloudlets in a time-shared manner within a VM. In order to ensure proper functioning, it is necessary for each VM to possess an individual instance of a CloudletScheduler. Four different scenarios with different cloudlet numbers and low/high heterogeneity are considered.

From the results of the experiments, it was observed that ABC based scheduler outperformed all the other scheduling policies considered in this research, inclusive of PSO. It had better min, max and avg makespan values for both the case studies evaluated in this study. Both PSO and ABC performed better than the baseline algorithms, FCFS and SJF, that supports the use of these scheduling mechanisms unsuitable in a fog cloud scenario. The other issue that came to light was the poor performance of PSO under scenario 2 in both the case studies. The max execution time of 550.3147 was 40% more than other policies and the standard deviation values

was 91.1793 for this scenario. This clearly outlines the incapacity of the PSO to minimize the makespan or manage resources efficiently when the heterogeneity of the network increases, making it unsuitable for a complex mix of edge/fog nodes, cloud devices etc. The proposed ABC clearly performed better and is more suitable to be deployed in fog/cloud scenarios.

7. Conclusion

In conclusion, the research topic on task scheduling in fog computing using PSO and ABC algorithms, simulated on CloudSim, addresses a critical challenge in the emerging field of fog computing. This research aimed to optimize task scheduling in dynamic fog environments to enhance resource utilization and minimize the makespan. The research technique included a systematic approach encompassing many stages, such as problem definition, algorithm adaption, simulation design, experimentation, and performance evaluation. The utilization of CloudSim allowed for the creation of a realistic fog computing environment, enabling the assessment of scheduling strategies under varying workloads, resource constraints, and dynamic conditions. The obtained results and performance evaluations provided valuable insights into the effectiveness of PSO and ABC for task scheduling in fog computing where it was realized that ABC based optimizer performed better under all considered scenarios. Future research scope may include hybridizing the metaheuristic approaches, to include to other objectives like cost minimization and energy efficiency or exploring the integration of machine learning for better adaptive and efficient dynamic scheduling solutions in fog computing.

References

Agarwal, M. and Srivastava, G.M.S., 2021. Opposition-based learning inspired particle swarm optimization (OPSO) scheme for task scheduling problem in cloud computing. *Journal of Ambient Intelligence and Humanized Computing*, *12*(10), pp.9855-9875.

Ben Alla, H., Ben Alla, S., Touhafi, A. and Ezzati, A., 2018. A novel task scheduling approach based on dynamic queues and hybrid meta-heuristic algorithms for cloud computing environment. *Cluster Computing*, *21*(4), pp.1797-1820.

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

Bittencourt, L.F., Diaz-Montes, J., Buyya, R., Rana, O.F. and Parashar, M., 2017. Mobility-aware application scheduling in fog computing. *IEEE Cloud Computing*, *4*(2), pp.26-35.

Chaudhary, D., Kumar, B. and Khanna, R., 2017. NPSO based cost optimization for load scheduling in cloud computing. In *Security in Computing and Communications: 5th International Symposium, SSCC 2017, Manipal, India, September 13–16, 2017, Proceedings 5* (pp. 109-121). Springer Singapore.

Dong, T., Xue, F., Xiao, C. and Li, J., 2020. Task scheduling based on deep reinforcement learning in a cloud manufacturing environment. *Concurrency and Computation: Practice and Experience*, *32*(11), p.e5654.

Durillo, J.J. and Prodan, R., 2014. Multi-objective workflow scheduling in Amazon EC2. *Cluster computing*, *17*, pp.169-189.

El-Boghdadi, H. and Rabie, A., 2019. Resource scheduling for offline cloud computing using deep reinforcement learning. *Int. J. Comput. Sci. Netw*, *19*, pp.342-356.

Elhady, G.F. and Tawfeek, M.A., 2015, December. A comparative study into swarm intelligence algorithms for dynamic tasks scheduling in cloud computing. In *2015 IEEE Seventh international conference on intelligent computing and information systems (ICICIS)* (pp. 362-369). IEEE.

Gabi, D., Ismail, A.S. and Dankolo, N.M., 2019, June. Minimized makespan based improved cat swarm optimization for efficient task scheduling in cloud datacenter. In *Proceedings of the 2019 3rd High Performance Computing and Cluster Technologies Conference* (pp. 16-20).

Gasmi, K., Dilek, S., Tosun, S. and Ozdemir, S., 2022. A survey on computation offloading and service placement in fog computing-based IoT. *The Journal of Supercomputing*, *78*(2), pp.1983-2014.

Hammoud, A., Mourad, A., Otrok, H., Wahab, O.A. and Harmanani, H., 2020. Cloud federation formation using genetic and evolutionary game theoretical models. *future generation computer systems*, *104*, pp.92-104.

He, H., Xu, G., Pang, S. and Zhao, Z., 2016. AMTS: Adaptive multi-objective task scheduling strategy in cloud computing. *China Communications*, *13*(4), pp.162-171.

Huang, X., Li, C., Chen, H. and An, D., 2020. Task scheduling in cloud computing using particle swarm optimization with time varying inertia weight strategies. *Cluster Computing*, *23*, pp.1137-1147.

Jiao, L., Friedman, R., Fu, X., Secci, S., Smoreda, Z. and Tschofenig, H., 2013. Cloud-based computation offloading for mobile devices: State of the art, challenges and opportunities. *2013 Future Network & Mobile Summit*, pp.1-11.

Kaur, N., Kumar, A. and Kumar, R., 2021. A systematic review on task scheduling in Fog computing: Taxonomy, tools, challenges, and future directions. *Concurrency and Computation: Practice and Experience*, *33*(21), p.e6432.

Li, D., Li, K., Liang, J. and Ouyang, A., 2019. A hybrid particle swarm optimization algorithm for load balancing of MDS on heterogeneous computing systems. *Neurocomputing*, *330*, pp.380-393.

Liu, N., Li, Z., Xu, J., Xu, Z., Lin, S., Qiu, Q., Tang, J. and Wang, Y., 2017, June. A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning. In *2017 IEEE 37th international conference on distributed computing systems (ICDCS)* (pp. 372-382). IEEE.

Long, X., Zhang, J., Zhou, K. and Jin, T., 2022. Dynamic self-learning artificial bee colony optimization algorithm for flexible job-shop scheduling problem with job insertion. *Processes*, *10*(3), p.571.

Luo, F., Yuan, Y., Ding, W. and Lu, H., 2018, October. An improved particle swarm optimization algorithm based on adaptive weight for task scheduling in cloud computing. In *Proceedings of the 2nd International Conference on Computer Science and Application Engineering* (pp. 1-5).

Madni, S.H.H., Latiff, M.S.A., Ali, J. and Abdulhamid, S.I.M., 2019. Multi-objective-oriented cuckoo search optimization-based resource scheduling algorithm for clouds. *Arabian Journal for Science and Engineering*, *44*, pp.3585-3602.

Mandal, T. and Acharyya, S., 2015, December. Optimal task scheduling in cloud computing environment: meta heuristic approaches. In *2015 2nd International Conference on Electrical Information and Communication Technologies (EICT)* (pp. 24-28). IEEE.

Mathew, T., Sekaran, K.C. and Jose, J., 2014, September. Study and analysis of various task scheduling algorithms in the cloud computing environment. In *2014 International conference on advances in computing, communications and informatics (ICACCI)* (pp. 658-664). IEEE.

Mouradian, C., Naboulsi, D., Yangui, S., Glitho, R.H., Morrow, M.J. and Polakos, P.A., 2017. A comprehensive survey on fog computing: State-of-the-art and research challenges. *IEEE communications surveys & tutorials*, *20*(1), pp.416-464.

Rahimikhanghah, A., Tajkey, M., Rezazadeh, B. and Rahmani, A.M., 2022. Resource scheduling methods in cloud and fog computing environments: a systematic literature review. *Cluster Computing*, pp.1-35.

Rajalakshmi, N.R., Arulkumaran, G. and Santhosh, J., 2019. Virtual Machine Consolidation for Performance and Energy Efficient Cloud Data Centre using Reinforcement Learning. *Int. J. Eng. Adv. Technol*, *8*, pp.78-85.

Raju, R., Babukarthik, R.G., Chandramohan, D., Dhavachelvan, P. and Vengattaraman, T., 2013, February. Minimizing the makespan using Hybrid algorithm for cloud computing. In *2013 3rd IEEE International Advance Computing Conference (IACC)* (pp. 957-962). IEEE.

Ramezani, F., Lu, J. and Hussain, F., 2013. Task scheduling optimization in cloud computing applying multi-objective particle swarm optimization. In *Service-Oriented Computing: 11th International Conference, ICSOC 2013, Berlin, Germany, December 2-5, 2013, Proceedings 11* (pp. 237-251). Springer Berlin Heidelberg.

Reddy, G.R.N. and Phanikumar, S., 2018. Multi Objective Task Scheduling Using Modified Ant Colony Optimization in Cloud Computing. *International Journal of Intelligent Engineering & Systems*, *11*(3).

Sharma, M. and Garg, R., 2020. HIGA: Harmony-inspired genetic algorithm for rack-aware energy-efficient task scheduling in cloud data centers. *Engineering Science and Technology, an International Journal*, *23*(1), pp.211-224.

Tawfeek, M.A. and Elhady, G.F., 2016. Hybrid algorithm based on swarm intelligence techniques for dynamic tasks scheduling in cloud computing. *International Journal of Intelligent Systems and Applications*, *8*(11), pp.61-69.

Tuli, S., Ilager, S., Ramamohanarao, K. and Buyya, R., 2020. Dynamic scheduling for stochastic edge-cloud computing environments using a3c learning and residual recurrent neural networks. *IEEE transactions on mobile computing*, *21*(3), pp.940-954.

Wang, T., Liu, Z., Chen, Y., Xu, Y. and Dai, X., 2014, August. Load balancing task scheduling based on genetic algorithm in cloud computing. In *2014 IEEE 12th international conference on dependable, autonomic and secure computing* (pp. 146-152). IEEE.

Wang, T., Wei, X., Tang, C. and Fan, J., 2018. Efficient multi-tasks scheduling algorithm in mobile cloud computing with time constraints. *Peer-to-Peer Networking and Applications*, *11*, pp.793-807.

Wu, X., Deng, M., Zhang, R., Zeng, B. and Zhou, S., 2013. A task scheduling algorithm based on QoS-driven in cloud computing. *Procedia Computer Science*, *17*, pp.1162-1169.

Yassa, S., Chelouah, R., Kadima, H. and Granado, B., 2013. Multi-objective approach for energy-aware workflow scheduling in cloud computing environments. *The Scientific World Journal*, *2013*.

Yuan, H. and Bi, J., 2019, October. Profit-Aware Spatial Task Scheduling in Distributed Green Clouds. In *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)* (pp. 421-426). IEEE.

Zade, B.M.H., Mansouri, N. and Javidi, M.M., 2021. Multi-objective scheduling technique based on hybrid hitchcock bird algorithm and fuzzy signature in cloud computing. *Engineering Applications of Artificial Intelligence*, *104*, p.104372.

Zhang, P. and Zhou, M., 2017. Dynamic cloud task scheduling based on a two-stage strategy. *IEEE Transactions on Automation Science and Engineering*, *15*(2), pp.772-783.

Zhou, Z., Chang, J., Hu, Z., Yu, J. and Li, F., 2018. A modified PSO algorithm for task scheduling optimization in cloud computing. *Concurrency and Computation: Practice and Experience*, *30*(24), p.e4970.