Journal of Intelligent Systems and Internet of Things
Vol. 12, No. 02, PP. 122-137, 2024
DOI: https://doi.org/10.54216/JISIoT.120209

Multi-Objective Evolutionary Algorithm to Optimize IoT Based Scheduling Problem Using (NSGA-II Algorithm)

Syed Mutiullah Hussaini[1,*], T. Abdul Razak[2], Muhammad Abid Jamil[3]

[1]Part-time Research Scholar, Jamal Mohammed College, Bharathidasan University, Trichy, Tamil Nadu State, India
[2]Associate Prof. in Computer Science Dept., Jamal Mohammed College, Bharathidasan University, Trichy, Tamil Nadu State, India
[3]Affiliation Department of Computer Science, Umm Al Qura University, Makkah, Saudi Arabia
Emails: smhussaini@gmail.com; abdul1964@gmail.com; majamil@uqu.edu.sa
[*]Corresponding Author: smhussaini@gmail.com

Abstract

Due to the continual advancements in the Internet of Things (IoT), which generate enormous volumes of data, the cloud computing infrastructure recently has received the most significance. to meet the demands made by the network of IoT devices. It is anticipated that the planned Fog computing system would constitute the next development in cloud computing. The optimal distribution of computing capacity to reduce processing times and operating costs is one of the tasks that fog computing confronts. In the IoT, fog computing is a decentralized computing approach that moves data storage and processing closer to the network's edge. This research article discusses a unique technique for lowering operating expenses and improving work scheduling in a cloud-fog environment. Non-dominated sorting genetic algorithm II (NSGA-II) is a proposal that is presented in this paper. Its purpose is to allocate service requests with the multi-objective of minimising finishing time and running cost. Determining the Pareto front that is associated with a group of perfect solutions, which are sometimes referred to as non-dominated solutions or Pareto sets, is the fundamental objective of the Pareto NSGA-II. There is a contradiction between the environmental and economic performances, which is shown by the Pareto set of sub-optimal solutions that are the consequence of the bi-objective issue.

Received: August 16, 2023 Revised: November 12, 2023 Accepted: April: 14, 2024

Keywords: Cloud computing; IoT devices; NSGA-II Algorithm; non-dominated; Optimization; Optimal distribution; Pareto set; Fog computing; multi-objective optimization

1. Introduction:

The IoT is a novel development in the communication and information technology sector. With IoT, in addition to more traditional smart devices like cellphones and laptops, Internet connectivity is expanded to a variety of things and devices (devices, machinery, autos, etc.) to put into practice many programs and services, such as traffic management, energy management, vehicle networks, and health and medical care. This creates a tremendous quantity of data, which must be processed, saved, and assessed in order to yield information that will serve users' needs. [1]

In addition, the quantity and variety of apps and services are expanding quickly, necessitating processing power that even the most recent smart gadgets can't provide. It is recognized that the cloud architecture's virtualization technologies are a crucial computing center that enables users to share and distribute resources dynamically. There's a wonderful opportunity to support IoT advancements. The constraints of the smart devices that are currently available, like their battery life, storage capacity, network resources, and processing speed, may be alleviated by shifting

operations that need a lot of labour and resources to a potent computer platform, like cloud computing, and allowing smart devices to handle jobs that are somewhat less complicated [2]

But when IoT and cloud computing are combined, other issues appear. The number of IoT devices that have been deployed is expected to expand from 15.4 billion in 2015 to 30.7 billion in 2020 and 75.4 billion in 2025, according to prognostications made by Information Handling Services (IHS) Arcade. Given the dramatic rise in the number of linked devices, the typical centralised processing parts of the Cloud architecture, which include the pooling of computing and storage resources and their storage in a number of different data centres, would not be able to encounter the necessities of IoT applications. There is a vast distance between IoT devices and the cloud, which is the primary cause. In addition to putting a burden on network capacity and performance, IoT devices that communicate with the cloud over the Internet will result in enormous amounts of data being communicated from latency. This is particularly true in areas that are close to obstacles. Since latency sensitivity is one of the characteristics of IoT applications, the transmission delay results in a reduced Quality of Service (QoS), which negatively impacts the user experience. Furthermore, because it is so expensive, IoT devices may not always have access to a continual connection to the Cloud. On the other hand, many network edge devices (such as routers, gateways, personal computers, workstations, etc.) now have an increasing number of processing, storage, and communication capabilities as a result of advancements in both hardware and software technology [3]. The effectiveness of our method was tested via experiments and compared to that of IBEA [4].

A novel concept of cloud computing called fog computing [5], first put forth by Cisco, can turn A networked interface into a distributed computing environment that can support IoT applications. The IoT devices that produce and use data may now benefit from the development of cloud computing, fog computing achieves its goal of bringing capabilities for processing and storage closer to consumers.

As an alternative to moving all processing activities to the cloud, fog computing, which is also referred to as fog servers or fog nodes, makes an effort to handle a portion of the workloads that are generated by applications on devices that are situated in close proximity to the network that users are connected to. These devices may be used anywhere there is network connectivity, such as in shopping centers, factories, electricity poles, railroad crossings, cars, and other locations. If they can do compute, storage, and networking, controllers, switches, routers, embedded servers, security cameras, etc. can all be classified as fog nodes. When resources are situated close to the network`s edge, where there is the least amount of latency in the process of data transmission to a computer node, the amount of time that the tasks take to transmit is maximised. Although the Fog Computing structure prioritises smaller operations or processing requests with minimal delays due to the limited processing capacity of a Fog Node, larger tasks and activities that can tolerate delays will still be routed to cloud tiers. In the end, cloud-fog computing would become a paradigm that complemented fog computing technology [6].

Table 1: Cloud computing and fog computing undergo comparisons.

Constraint	**Fog Computing**	**Cloud Computing**
Computing Capabilities	Reduces the amount of data sent to Cloud Computing.	No reduction in data while sending or transforming data.
Communication with Devices	From the edge via various conventions and rules, One step.	From a distance via Internet, Multiple stages.
Architecture	Distributed	Centralized
Latency	Low	High
Location of Server Nodes	Edge of the local network	Within the Internet
Data Processing	Local to the informational source in short-term time	Remote from the source of information in long-term time
Response time	High	Low

Security	Increased security, Definable	Less secure, Undefined
Server Nodes	Very large nodes with limited storage.	Few nodes with scalable storage.
Setting for Work	Outdoor or indoor	Warehouse-size with air conditioning systems

Chiang and Zhang [7] claim that fog computing can assist in addressing difficulties related to the IoT.

- Latency constraints: Fog computing is typically the greatest solution for addressing the severe timing requests of numerous IoT systems. This includes doing operations such as preserving and analysing data as well as other latency-limiting jobs near end users.
- Constraints on network bandwidth: Through the usage of fog computing, data may be processed in a hierarchical fashion from end users to the cloud. This allows for a trade-off to be made between the requirements of the application and the resources that are available for computing and networking. Furthermore, by optimising the use of data, a reduced amount of data will be essential to be transmitted to the cloud.
- Devices with restricted resources: Fog is able to manage resource-intensive tasks in lieu of resource-limited devices in situations when resources can't be sent to the cloud. This results in a reduction in the intricacy, maintenance costs, and usage of energy of these devices.
- Uninterrupted services with inconsistent cloud access: A fog system is able to function independently and deliver continuous services, even when there is only intermittent network access to the cloud.
- IoT security difficulties: The following can be done using a fog system: (1) serving as stand-ins for devices with low resources, via which software and authentication information may be updated and controlled; (2) carrying out a large number of security procedures on behalf of the devices with low resources in order to compensate for the absence of security features on these devices; (3) keeping an eye on the safekeeping prestige of devices in the immediate vicinity; and (4) making use of the context and information obtained in order to swiftly detect potential dangers.

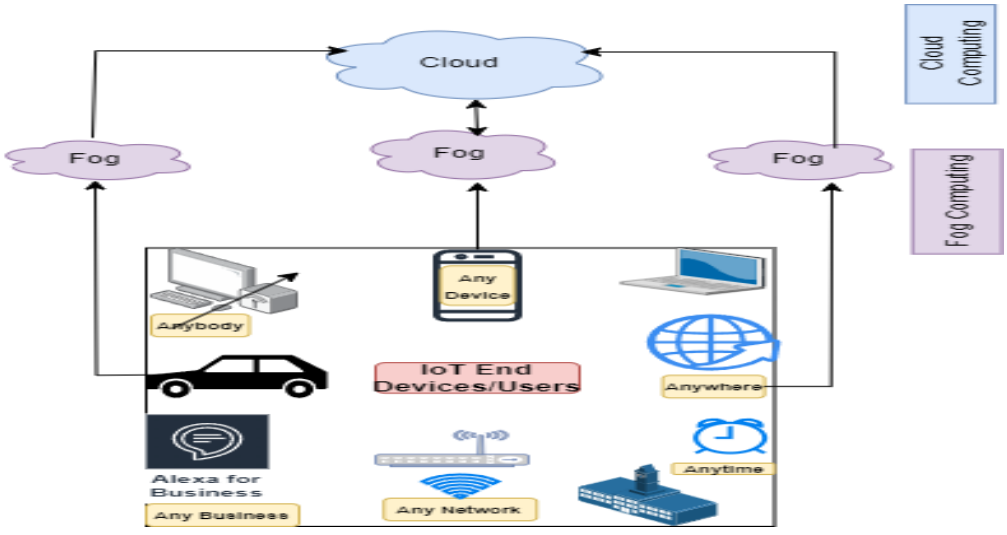

Figure 1: Diagram of Cloud computing and Fog Computing Architecture

In this analysis, we focus on the scheduler. The use of Time-Cost Aware Scheduling (TCaS) is advised as a solution to address this problem. The TCaS technique's fundamental goal is to effectively manage implementation time with costs to finish a range of jobs in the Cloud-Fog environment. Furthermore, this technique is adaptable enough to satisfy various users' expectations, such as those who want to emphasize job completion now against those who want to

accomplish their duties on a limited income. The following is the structure of the remainder of the essay: The relevant Section 2 displays the works. Section 3 provides a mathematical representation and description of the job scheduling issue in the context of cloud-fog computing. Our suggested NSGA-II algorithm job scheduling problem is presented in full in Section 4. Section 5 displays Crossover operator. Section 6 presents the experimental findings. Section 7 brings the article to a close and discusses potential future efforts.

2. Related Work

2.1 IoT with Fog Computing

IoT applications are generating several challenges for the centralized cloud architecture. IoT, for instance, intelligent traffic signals [5], cannot handle real-time and low latency applications like linked vehicles [6], etc. These problems can be resolved through fog computing. the distinctions between cloud computing and fog computing. Fog computing, in the opinion of Chiang, M.; Zhang, T. [7], can aid in addressing issues with the Internet of Things.

Survey papers have been written about several facets of fog computing. Shi, Y. [8], for instance, looked at the crucial components for healthcare applications of fog computing. In addition, Yi, S.[9] 's research on fog computing included a discussion of different fog computing application scenarios and potential issues that might occur while putting these systems into place. The suggested architecture places central Fog services under a software-defined resource management layer [10]. This platform provides a cloud-based interface designed to prevent autonomous behaviour of Fog colony colonies. Cloud-based infrastructure is used to analyse, regulate, and monitor fog cells. Bonomi F. [11] investigated the incorporation of IoT with Fog computing by assessing fundamental components of Fog computing and the ways in which Fog enhances and expands Cloud computing. A hierarchically distributed Fog design was also suggested. To evaluate the qualities of their design, they offered examples for a wind generator and an automated traffic signal system.

A paradigm for comprehending, modeling delays, and assessing in IoT-Fog-Cloud systems was put forth by Yousefpour, A. [12]. They suggested a delay-minimizing Fog nodes policy to reduce service delay for IoT nodes. By sharing the burden, the suggested method adopts communication across Fog to decrease service delays. For computation dumping, the strategy takes into account queue lengths, as well as a variety of request kinds and processing durations. Furthermore, in order to analyze service delays in IoT-Fog-Cloud situations, the authors developed a mathematical model, which was supported by accurate simulation testing suggested regulations.

The security and privacy concerns brought on by combining fog computing with the IoT were examined by Lee, K. [13]. They claimed that implementing the IoT with fog creates a number of security risks. The requirement for creating a safe Fog computing platform employing several safety technologies was also emphasized. They also reviewed existing security methods that can be effective to secure the IoT with Fog.

In addition, Hong, K. [14] developed a Modular Fog (MF) in an organizational computing framework that can deploy applications for the IoT across a variety of devices, from edge devices to the Cloud. The MF blends components in parent-child connections, where parent nodes manage data from child nodes, using a dynamic node discovery mechanism. MFs are great for Web of Things applications since they can gather and assess information locally on end-client gadgets.

A framework for the classification of the fog computing context and information were supplied by Mahmud, R. [15] on its difficulties and distinguishing characteristics. They highlighted the variations between Mobile edge computing, Mobile cloud computing, Fog computing, Cloud computing, and Edge computing. They looked at networking setups, different Fog computing metrics, and Fog node settings.

2.2. Fog Computing Task Scheduling Algorithms

Over the course of some time, the issue of task scheduling in cloud computing has garnered a lot of attention.. Numerous studies have recently been conducted on this subject since the idea of fog computing was first put forth. The key concepts, the approach, the criterion for improvement, and the limits of earlier comparable works are summarized in the following Table 2.

Table 2: An overview of relevant literature on job scheduling methods on Fog computing.

Article	Concepts	Development Measures	Drawback
Abdi, S. (2014) [18]	Particle swarm optimization	Completion of job	In Cloud environment
Oueis, J.(2015) [19]	Heuristic	Job delay, power usage, and user pleasure	complexity as user requests grow in quantity
Guo,X. (2016) [20]	Decision problems using Markov	Power usage and latency	An uninterrupted-time queuing mechanism
Deng, R. (2016) [21]	Numerical non-linearities, Hungarian approach	Power usage and latency	Good for centralized systems, difficult to use in fog computing
Ningning,S. (2016) [22]	Graph split and spanning tree minimization	Execution time	complexity as the quantity of user requests rises
Gu, L. (2017) [23]	Heuristic based on two-phase linear programming	Application for wireless network, communication cost	Lack of capability to offload computation
Bitam, S. (2017) [24]	Bee Life Algorithm	Storage and execution time	Tiny dataset, limited to the fog environment
Binh, H.T.T.(2018) [25]	Genetic algorithm	Execution time, processing cost	Small dataset

Job scheduling utilizing adjusted PSO calculation in distributed computing climate by Abdi, S. [16] fostered a Modified Particle Swarm Optimization (MPSO). They differentiated it to two notable heuristic strategies, specifically GA and PSO, to decide the effectiveness of occupation planning for the setting of distributed computing. They concentrated on speeding up task completion. To improve the initial population, the PSO and the smallest job to fastest processor method (SJFP) were combined. Although it has only been evaluated in a cloud context, the MPSO algorithm outperforms both regular PSO and GA in terms of results.

Oueis, J. [17] addressed load balancing in fog computing with the goal of enhancing the quality of user experience (QoE). First, they divided up the available local computer power into smaller groups. Finally, in fog computing, clusters are generated for every user's request based on a specific optimisation aim of arrival time, latency, and other factors. This is done in order to maximise efficiency. With just a little amount of computer equipment, they were able to accomplish satisfactory results; but, if the Fog computing infrastructure is significantly expanded, the procedure may become more complicated.

Guo,X.'s [18] main goal was to reduce latency in edge clouds. They put forth the Cloud-Fog computing task allocation dilemma. Nonetheless, they employed a basic model to calculate delay and power usage. Specifically, base stations

receive tasks from end-user devices. The base stations are responsible for receiving tasks and then transmitting them to the edge cloud servers so that they may be completed. The end consumers are then given access to the results.

When assigning work in the Cloud–Fog environment, Deng, R [19] used the optimization strategies already in place to address three different sub-problems with the goal of minimizing power usage and delay. First, they discovered the ideal relationship between fog computing latency and power usage.

For fog computing, Ningning, S [20] suggested a graph-based load balancing method. Based on the resources that are required, it is feasible for tasks to be assigned to either one fog node or numerous fog nodes. The network of fog nodes is shown as a graph that is not directed specifically. This procedure has an effect on the speed at which jobs and tasks are completed. The most significant drawback of this approach is that it is not the most effective solution for dynamic load balancing fog computing. This is due to the fact that the graph often reorganizes itself to accommodate alterations in fog computing.

Within the framework of medical cyber-physical systems that are made possible by fog computing, Gu, L. [21] investigated the distribution of jobs, the affiliation of base stations, and the placement of virtual machines. To guarantee Qos, they decreased the total cost. Having said that, the model that has been proposed is constructed on the architecture of cellular networks; thus, it does not extend into situations that occur in the Cloud–Fog computing conditions. The Cloud entity is not included in the model, and it is suppose that fog nodes are located in the same location as the base station. Furthermore, fog nodes do not have the ability to offload computation.

Bitam,S. [22] suggested utilizing the Bee Life Algorithm (BLA) as a work scheduling technique. Two primary goals of the essay are memory and execution time. The approach is only tested on tiny datasets, and the relationship with Cloud data center is not mentioned in this paper.Bitam,S. [22] suggested utilizing the Bee Life Algorithm (BLA) as a work scheduling technique. Two primary goals of the essay are memory and execution time. The approach is only tested on tiny datasets, and the relationship with Cloud data center is not mentioned in this paper.

In a Cloud-Fog computing conditions, Binh, H.T.T. [23] suggested a job scheduling technique based on Genetic Algorithms (GA). This approach seeks to achieve a favorable time or cost trade-off.

The overwhelming bulk of the research that is pertinent implements methods only in fog or cloud computing conditions, which are systems in which processing nodes have similar power requirements. Workloads are not distributed equally between processing nodes in the hybrid Cloud-Fog system, in contrast to where they are distributed in Cloud or Fog computing. The reason for this is because Fog nodes and Cloud nodes are drastically distinct from one another. This may result in a lot of the traditional cloud-based scheduling systems not functioning as well as they should. The significant contribution that was made as a consequence of this research was the creation of the TCaS algorithm, which is an approach that is based on evolutionary algorithms and aims to reach the highest possible trade-off between the amount of time its execution takes and the amount of funds it costs to process. In addition, we find a solution to the issue of work scheduling in the Cloud-Fog computing system by utilising a variety of different evolutionary algorithms, the most notable of which are the Bee Life Algorithm or method (BLA) and the Particle Swarm Optimisation (PSO) algorithm. Additionally, we use the straightforward Round Robin (RR) technique to illustrate the effectiveness of the method [5].

3. Problems with Job Scheduling

3.1. System Architecture Design

The definition of the cloud job scheduling problem is the scheduling and practical allocation of diverse jobs to numerous virtual machines (VMs) so that all jobs are completed in a brief execution period. Assume about the cloud system (CS), which consists of N_{vm} virtual machines (VMs) on each PM and N_{pm} physical machines (PM).

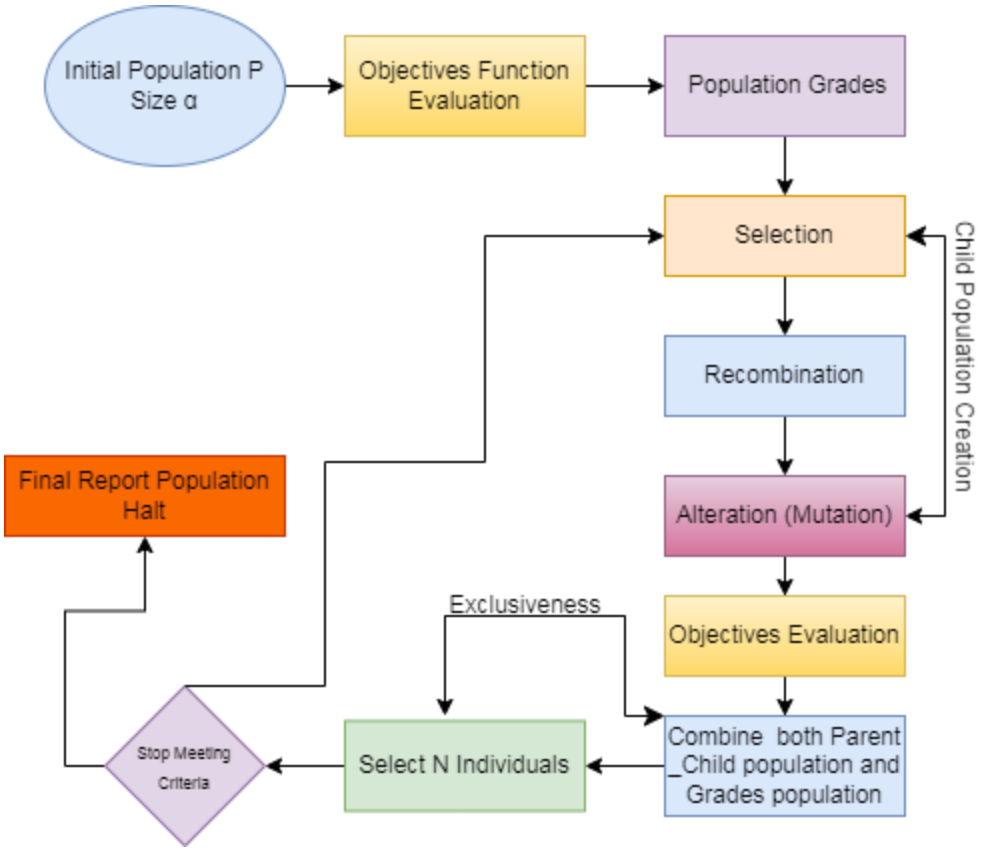

Figure 2: NSGAII Algorithm Work Flow

The next generation is chosen using NSGA-II based on non-dominated-sorting and crowding distance comparison after offspring are produced utilizing a certain kind of crossover and mutation. Box-bounded multiple-objective optimization may be done using the version that was implemented in pagmo. NSGA-II is a reliable multi-objective algorithm, widely utilized in many real-world situations. Although the NSGA-II technique is now seen as being out of date, it still has a lot of usefulness, if not only as a reliable benchmark.

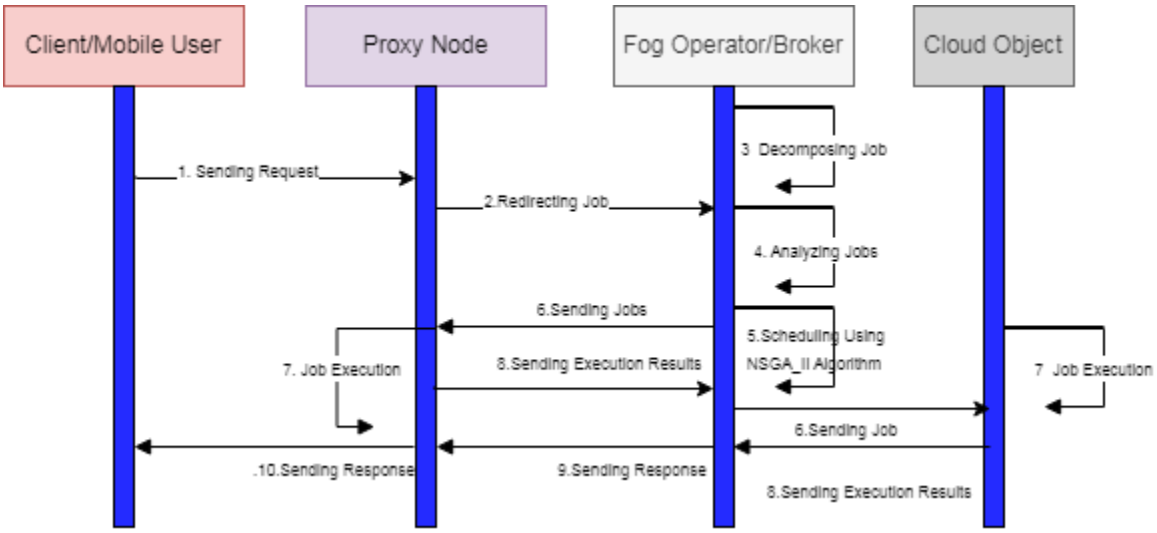

Figure 3: The Cloud–Fog system operation

For the sake of system performance, the TCaS algorithm was created. It is implemented on the Fog broker and seeks to find the optimal job execution schedule by reaching cost and time effectiveness. The sequential execution of our system model is decomposed in Figure 3.

During the first stage (step 1), a mobile user will send in a request, which will be handled by the Fog (Proxy) node that we have connected. It is instantly made accessible to the Fog broker (Step 2), which is the request (or task/job). Step 3 involves the process of breaking down each work into a list of tasks that may be carried out in a distributed system. Following this step, Step 4 involves the calculation of the anticipated number of instructions and the utilisation of resources. Step 5 involves the execution of a scheduling algorithm, which is managed by the Fog broker, which is accountable for handling all of the information about tasks and nodes.

Step 6 involves the transmission of tasks to the Fog and Cloud nodes that are the more suitable for them., which comes after the output. At the seventh step, it is the responsibility of every node to finish all of the jobs that have been assigned to it. After that, it is required to forward the finished tasks back to the fog broker at the step 8. Following the completion of all tasks, the Fog broker will integrate the results of the work (Step 9). Step 10 involves using the connected fog node to deliver the response to the mobile user next.

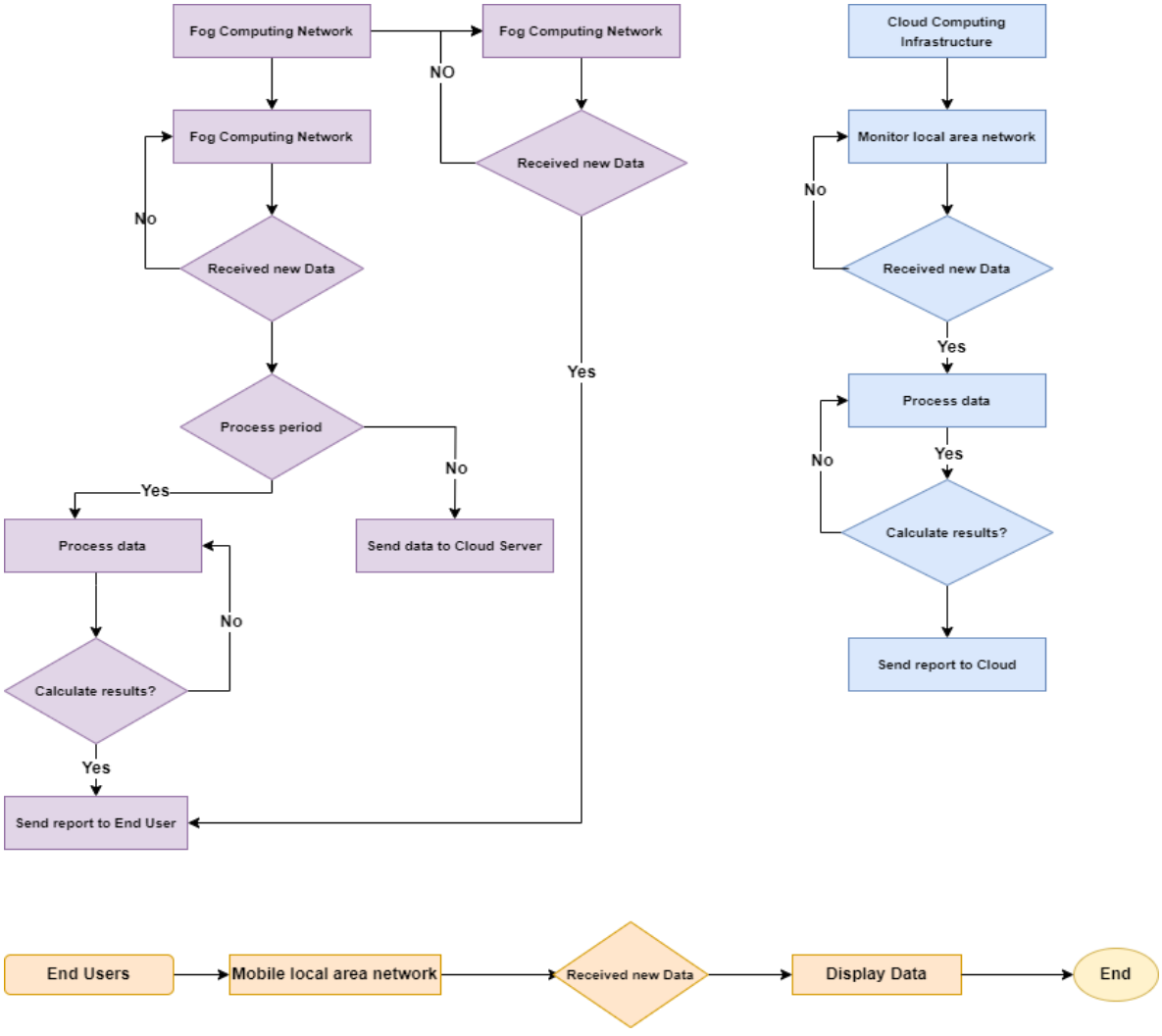

Figure 4: Flow chart of all processes stirring in the cloud/fog side operation system.

3.2. Problem Explanation

Requests from IoT based Bag-of-Jobs (BoJ) applications are broken down into manageable, independent jobs, whenever the Cloud-Fog computing framework transmits them to the Fog layer for being handled. All job's attributes include the quantity of directions, the quantity of memory required, and the dimension of the input and output files.

Assuming that J_c stands for the cth job , the system receives a collection of n distinct jobs each time, as follows: As stated in our previous paper.

$$J = \{J_1, J_2, J_3, \dots, J_n\} \quad (1)$$

The Cloud-Fog computing framework is made up of Cloud networks and Foggy nodes, which are processors with the same CPU frequency, CPU usage charge, memory usage cost, and consumption of bandwidth price as each other. Although cloud nodes often have higher power than fog nodes, their use is more expensive. The collection of n processors known as the system's c Cloud nodes and f Fog nodes includes these nodes.

$K(K = K_{cloud\ U} K_{Fog})$, which is expressed as

$$K = \{K_1, K_2, K_3, \dots, K_n\}, \quad (2)$$

where the i^{th} processing node is displayed by Ki.

Every Job J_c

The processor ($J_c \in J_c\ jobs$) is given $K_i (K_i \in Nodes)$, which is denoted as J_c^i.

$$K_i Jobs = \{J_x^i, J_y^i, \dots, J_z^i\} \quad (3)$$

In a Cloud-Fog computing system, the job scheduling issue could be stated as set searching.

$$NodeJobs = \{J_1^a, J_2^b, J_3^c, \dots, J_n^p\} \quad (4)$$

The execution time (EXT) required by a node Ki to perform all jobs assigned for a collection of jobs is: (Execution Time as to be objective -1)

Where $ExeTime(J_i)$

$$EXT(K_i) = \sum J_c^i \in K_i\ Jobs \qquad ExeTime(J_c^i) = \frac{\sum J_c^I \in K_i Jobs\ length(J_c)}{CPUrate(K_i)} \quad (5)$$

Where ExeTime(J_k^i) is the time at which node K_i processed J_k. which is determined by:

$$ExeTime(J_c^i) = \frac{length(J_c^i)}{CPUrate(K_i)} \quad (6)$$

, where $length(J_k^i)$ is the amount of instructions in task $J_k$and $CPUrate(K_i)$ is the node $K_{i`s}$ CPU rate.

This is reliant upon clock rate, center count, parallelism at the direction/instruction level, and so forth. Makespan is the whole period of time required for the framework to play out each work, determined from the time a solicitation is gotten until the last undertaking has been done or the last machine is utilized.

The formula determines Makespan.

$$Makespan = Max[EXT((K_i)] \quad (7)$$
$$1 \le i \le m$$

Let Minimum Makespan represent the shortest amount of time necessary for the system to complete all jobs or the shorter limit of Makespan. Minimal Makespan will be found and calculated by assuming that all nodes do their allocated tasks concurrently.

$$MinimalMakespan = EXT(K_i) = \cdots = EXT((K_m)$$

Thus,

$$MinimalMakespan = \frac{\sum 1 \le k \le n\ length(J_c)}{\sum 1 \le i \le n\ CPUrate(K_i)} \quad (8)$$

Payment is required for processing, memory, and bandwidth costs each time a task is completed in the Cloud-Fog system. The projected cost for node K_i processing task J_c is given as follows:

$$Cost(J_c^i) = cp(J_c^i)cm(J_c^i) + cb(J_c^i) \quad (9)$$

Each cost is computed using Equation (5) as follows.
As determined by processing cost: (Processing Cost Objective -------2)

$$cp(J_c^i) = c1 * ExeTime(J_c^i) \quad (10)$$

where $ExeTime(J_c^i)$ is defined as Equation and c1 is the cost of CPU consumption per time unit in node K_i (6).
Given that Mem(J_c) is the amount of memory needed by task J_k and c2 is the cost of memory usage per data unit in node Ki, the following is the memory usage cost:

$$cm(J_c^i) = c2 * Mem(J_c^i) \quad (11)$$

Processed task J_k in node The total of the input and output file sizes, or Bw(J_k), is the amount of bandwidth that K_i requires. Let c3 represent the price of bandwidth usage per data unit; it is defined as follows:

$$cb(J_c^i) = c3 * Bw(J_c^c) \quad (12)$$

The following is a calculation of the total cost for all jobs to be finished in the Cloud-Fog system:

$$TotalCost = \sum_{J_c^i \in NodeTasks} Cost(J_c^i) \quad (13)$$

When every job is allocated to the least expensive node, MinTotalCost—the least cost required for a collection of jobs J to be accomplished in the Cloud–Fog system—is acquired. MinCost(J_c), the node that processes job J_c at the lowest cost, may be easily found given the information about each node.

4. NSGA_II for Job Scheduling Problem

4.1 Population Initialization

Supposing that a population of N particles exists, each particle's position in the initial population is initially produced at random, ensuring that particles are dispersed throughout the search space. Additionally, enabling dynamic exploration, the velocity matrices of the particles are initialized stochastic-ally.

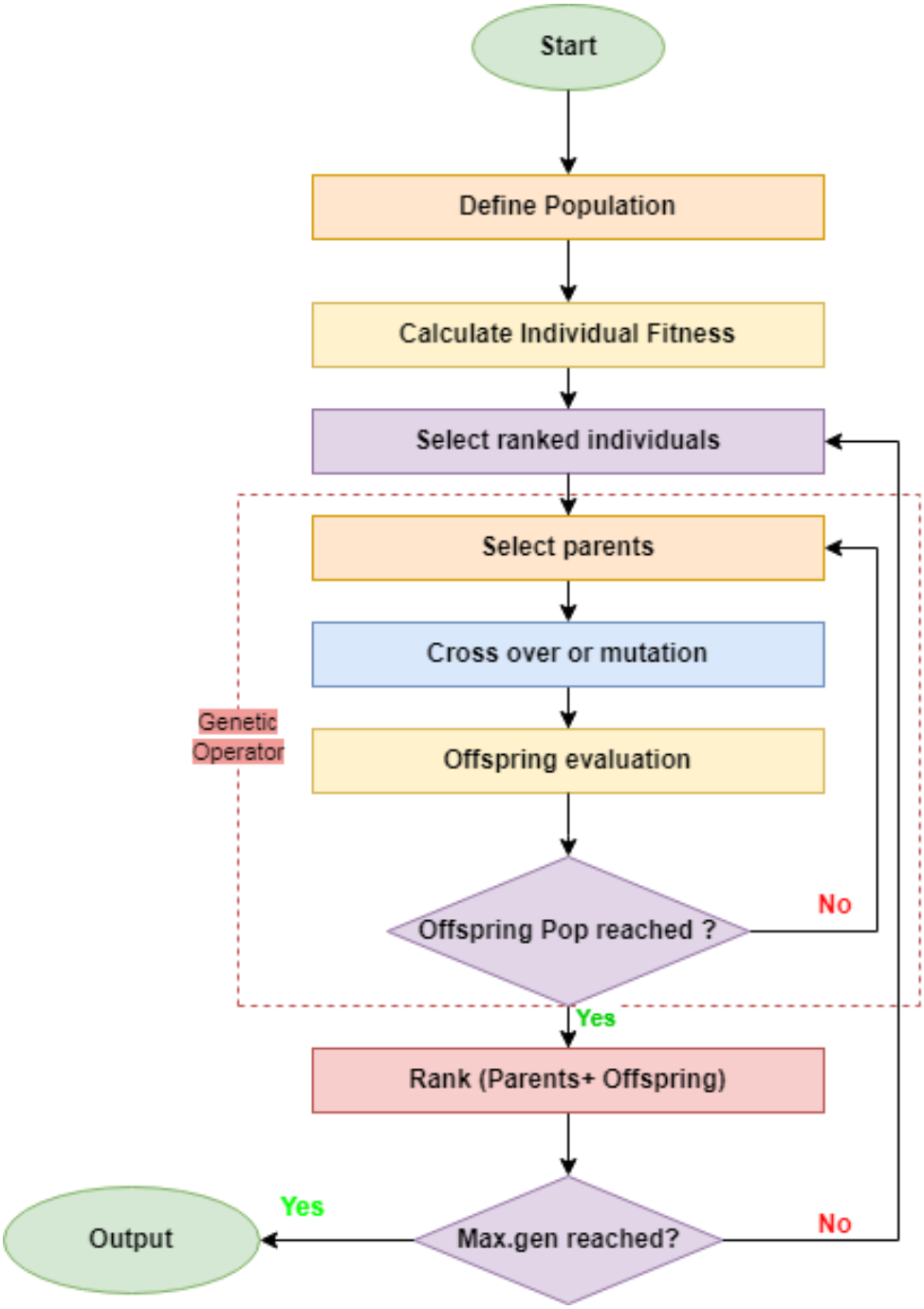

Figure 5: NSGA_II Proposed Methodology

Population is sorted into different levels by using the NSGA II to first arbitrarily generate a set of non-dominated preferred sizes and then compute the objective functions for every individual non-dominated sorting technique. Next, Population who are organized into the same level are sorted using the crowding distance method.

NSGA_II_ Step by Step Process Algorithm

Input: Population P

1 Define *population: the number of population;*

2 Create random *population P*

3 Calculate objectives values

4 Allocate Rank

5 Create Child Population for *size P*

6 for i=1 :*Max* **do**

7 **for** *each parent and child Є P* **do**

8 *Allocate Rank*

9 Create sets of no dominated *results*

10 *Cross over and mutation*

11 *Loop based on present result to next generation*

12 **end for**

13 Optimal points on the lower front with high distance

14 Create *next generation*

15 end for

Output : *Child*

A function that measures fitness is used to evaluate a particle; its value of fitness displays the quality of the outcome that the element represents as well as its impact on the population. Higher fitness values result in better solutions.

5. Genetic Operators

The procedure of dual-point crossover is utilised to produce offspring that inherit beneficial genetic factors from their parents. This is accomplished by encoding chromosomes as an array of numbers. In Figure 6, this process is illustrated. A new person is produced by randomly selecting two crossover locations, in which the first parent provides the second parent with the middle gene segment while maintaining the integrity of the other gene.

The success of the algorithm in the crossover procedure for the inhabitants is impacted by the selection of parents. Everyone has a crossover frequency. The population's second parent is then chosen at random to contribute in the crossover procedure after each member of the population is told of their likelihood of becoming the population's first parent. Each individual has a crossover rate. The population's second parent is then chosen at random to take part in the crossover procedure after each person is told of their likelihood of being the population's first parent. With this kind of selection, superior individuals with greater fitness values are more likely to be chosen as parents, ensuring that superior gene segments are more probable to be kept in the following generation.

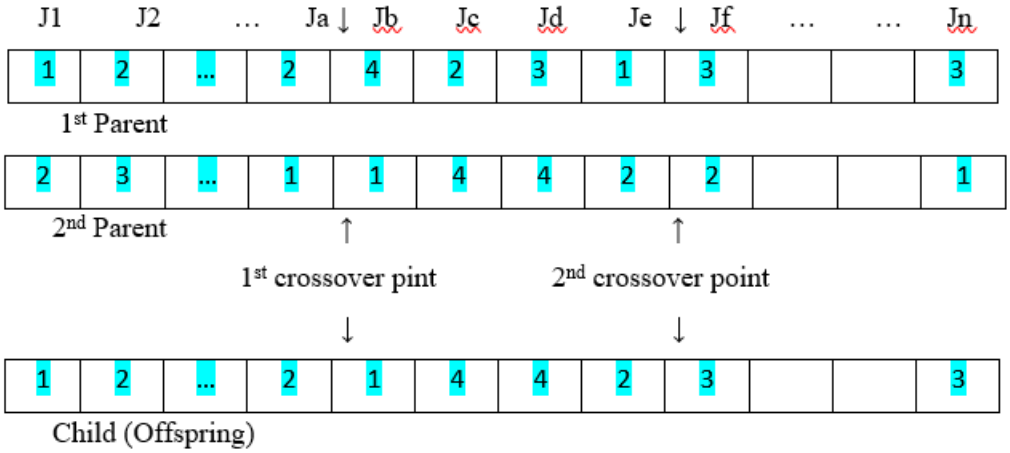

Figure 6: Two-point Chromosome Encoding crossover operator.

To create two new children, a two-point crossover operator randomly chooses two crossover spots inside a chromosome and then switches the two parent chromosomes between these points. Where J1, J2 a set of independent jobs etc.

6. Performance Assessment

In this part, we will give the evaluation of the NSGA_II method that was developed based on the data sets that were obtained from the findings.

6.1 Experimental Settings & Results

In terms of resource utilisation costs and processing factors, cloud and fog nodes are distinct from one another. As a depiction of every node's particular processing capability, we most likely examined the processing rate of every node, together with the expenses of its CPU, memory, and bandwidth usage. A variety of processing nodes that were used to construct the Cloud-Fog Structure are described in Table 1, which may be found below. Routers, workstations, gateways, and personal computers are examples of several types of fog layer nodes that have a limited capability for computation. Within the cloud layer, tasks are managed by servers or virtual machines that are located in data centres that are designed for high performance. As a consequence of this, in comparison to fog nodes, cloud nodes process information at a substantially faster rate. However, compared to the Fog, the Cloud has a larger resource consumption cost.

Table 3: Cloud-Fog Structure's characteristics.

Parameter(s)	**Fog Setting**
CPU usage cost	[0.1,0.4]
CPU rate	[500,1500]
Memory Usage Cost	[0.01,0.03]
Bandwidth usage cost	[0.01,0.03]
Number of Nodes	10

It is the responsibility of the Cloud-Fog structure to fulfil all of the needs that have been expressed by Users. A list of tasks is compiled from each request, which is then examined, and the quantity of resources that are anticipated to be required for each work is established. According to the amount of labour every single request requires, the number of tasks that are included in the collection could be rather varied from one another. As a result, 10 datasets containing between 40 and 500 jobs were made to take part in our experiment.

Table 4: NSGA_II Parameter settings

Parameter(s)	**Values**
Crossover rate	60%
Mutation rate	60%
Number of generations	500
Population size	250

Table 4 shows the best parameters setting for the proposed problem. Meanwhile our suggested NSGA_II algorithm is based on Genetic Algorithm.

6.2 Testing Results and Discussion

Numerous tests were carried out with two separate objectives to evaluate our processes. The first step consisted of a variety of tasks being carried out locally inside a network of linked Fog nodes. The Fog layer, in combination with a bunch of cloud nodes, was responsible for processing user requests throughout the second stage of progress.

6.2.1 Objective 1: Job Scheduling in Fog system environment

To accommodate user requirements in a particular area, we initially configured a Fog layer made up of 10 Fog nodes that were linked to one another.

The NSGA_II algorithm's scheduling time and processing costs are shown in the following Table 5. Make-span and total costs were the two key factors that contributed to the function of fitness. As a result, the NSGA_II algorithm showed superiority on practically all datasets in terms of both time and cost.

The schedule of the NSGA_II algorithm was beaten by our suggested solution in terms of execution time.

Table 5: Make-span and total cost for NSGA_II

	Make-Span (s)	**Cost**
Number of Jobs	**NSGA_II**	**NSGA_II**
60	192.40*	735.58
90	395.55	1545.32
110	609.65	2365.47
150	822.67	3275.15
210	945.85	3835.35
250	1,285.95	4988.95

6.2.2. Objective 2 : Job scheduling in Cloud Fog-System

The entire Cloud-Fog system, which consisted of ten Fog nodes and three Cloud nodes that were separated from one another via a distance between them, was the subject of research that was carried out within the framework of this specific incidence. As opposed to goal 1, the attribute values of fog nodes were consistently the same, and it seemed as if the responsibilities were distributed equitably among them. The processing nodes in this instance were split into

two groups. More jobs were processed by cloud nodes than by Fog nodes since they were significantly more powerful. Therefore, it was more challenging to identify the best answer.

7. Future work and Conclusions

Within the context of the Cloud-Fog computing paradigm, the difficulty of work scheduling for IoT applications was the primary focus of the current study. The performance of the NSGA_II algorithm, which is based on an evolutionary process, was evaluated with regard to the minimization of two different goals that were being considered. In particular, the NSGA_II approach, which has been proposed, is accountable for the generation of the trade-off among time and cost execution. This is particularly true with regard to the acquisition of significantly shorter scheduling durations, such as those discussed in [24] [25]. Also, the suggested algorithm might adapt to users' needs for cost effectiveness or high performance processing.

In order to solve scheduling issues, we will continue to research, improve, and use additional algorithms, notably evolutionary algorithms. In order to satisfy users, we also intend to broaden the scheduling problem by concentrating on maximizing a variety of other objectives, including time, transmission costs, computing resources, and energy use. For increased practicality, the limitations of budget, deadline, and resources might be included.

Acknowledgement

The authors extend their appreciation to the Deputyship for Research & Innovation, Ministry of Education in Saudi Arabia for funding this research work through the project number: IFP22UQU4320619DSR113.

References

[1] Research on intelligent workshop resource scheduling method based on improved NSGA-II algorithm Minghai Yuan Volume 71, October 2021, 102141.

[2] Modelling and scheduling multi-objective flow shop problems with interfering jobs
Appl. Soft Comput. M. Torkashvand *et al.* (2017)

[3] Raspberry Pi as a Platform for the Internet of Things Projects: Experiences and Lessons Stan Kurkovsky, Chad Williams, ITiCSE '17, July 3-5, 2017

[4] Syed Muitullah Hussaini, 'Optimizing scheduling problem IoT based cloud computing environment using EAs (IBEA algorithm)', Journal of Data Acquisition and Processing Vol. 38 (3) 2023, 3634-3645.

[5] Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are; Cisco White Paper; 2015.

[6] Peter, N. Fog computing and its real time applications. Int. J. Emerg. Technol. Adv. Eng. 2015, 5, 266–269.

[7] Chiang, M.; Zhang, T. Fog and IoT: An overview of research opportunities. IEEE Internet Things J. 2016, 3, 854–864.

[8] Shi, Y.; Ding, G.; Wang, H.; Roman, H.E.; Lu, S. The fog computing service for healthcare. In Proceedings of the 2015 2nd International Symposium on Future Information and Communication Technologies for Ubiquitous HealthCare (Ubi-HealthTech), IEEE, Beijing, China, 28–30 May 2015; pp. 1–5.

[9] Yi, S.; Li, C.; Li, Q. A survey of fog computing: Concepts, applications and issues. In Proceedings of the 2015 Workshop on Mobile Big Data, Santa Clara, CA, USA, 29 October–1 November 2015; pp. 37–42.

[10] Suárez-Albela, M.; Fernández-Caramés, T.; Fraga-Lamas, P.; Castedo, L. A practical evaluation of a high-security energy-efficient gateway for IoT fog computing applications. Sensors 2017, 17, 1978.

[11] Bonomi, F.; Milito, R.; Natarajan, P.; Zhu, J. Fog computing: A platform for internet of things and analytics. In Big Data and Internet of Things: A Roadmap for Smart Environments; Springer: Berlin, Germany, 2014; pp. 169–186.

[12] Yousefpour, A.; Ishigaki, G.; Jue, J.P. Fog computing: Towards minimizing delay in the internet of things. In Proceedings of the 2017 IEEE International Conference on Edge Computing (EDGE), Honolulu, HI, USA, 25–30 June 2017; pp. 17–24.

[13] Lee, K.; Kim, D.; Ha, D.; Rajput, U.; Oh, H. On security and privacy issues of fog computing supported Internet of Things environment. In Proceedings of the 2015 6th International Conference on the Network of the Future (NOF), Montreal, QC, Canada, 30 September–2 October 2015; pp. 1–3.

[14] Hong, K.; Lillethun, D.; Ramachandran, U.; Ottenwälder, B.; Koldehofe, B. Mobile fog: A programming model for large-scale applications on the internet of things. In Proceedings of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing, Hong Kong, China, 2013; pp. 15–20.

[15] Mahmud, R.; Kotagiri, R.; Buyya, R. Fog computing: A taxonomy, survey and future directions. In Internet of Everything; Springer: Singapore, 2018; pp. 103–130.

[16] Abdi, S.; Motamedi, S.A.; Sharifian, S. Task scheduling using modified PSO algorithm in cloud computing environment. In Proceedings of the International Conference on Machine Learning, Electrical and Mechanical Engineering, Dubai, UAE, 8–9 January 2014; pp. 8–9.

[17] Oueis, J.; Strinati, E.C.; Barbarossa, S. The fog balancing: Load distribution for small cell cloud computing. In Proceedings of the 2015 IEEE 81st Vehicular Technology Conference (VTC Spring), Glasgow, UK, 11–14 May 2015; pp. 1–6.

[18] Li, D.; Sun, X. Nonlinear Integer Programming; Springer Science & Business Media: Berlin, Germany, 2006; Volume 84.

[19] Vishwesh Nagamalla, J.Raj karkee, Ravi Kumar Sanapala, Integrating Predictive Big Data Analytics with Behavioral Machine Learning Models for Proactive Threat Intelligence in Industrial IoT Cybersecurity, Journal of International Journal of Wireless and Ad Hoc Communication, Vol. 7 , No. 2 , (2023) : 08-24 (Doi : https://doi.org/10.54216/IJWAC.070201)

[20] Ningning, S.; Chao, G.; Xingshuo, A.; Qiang, Z. Fog computing dynamic load balancing mechanism based on graph repartitioning. China Commun. 2016, 13, 156–164.

[21] Deng, R.; Lu, R.; Lai, C.; Luan, T.H.; Liang, H. Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption. IEEE Internet Things J. 2016, 3, 1171–1181.

[22] Ningning, S.; Chao, G.; Xingshuo, A.; Qiang, Z. Fog computing dynamic load balancing mechanism based on graph repartitioning. China Commun. 2016, 13, 156–164.

[23] Gu, L.; Zeng, D.; Guo, S.; Barnawi, A.; Xiang, Y. Cost efficient resource management in fog computing supported medical cyber-physical system. IEEE Trans. Emerg. Top. Comput. 2017, 5, 108–119.

[24] Bitam, S.; Zeadally, S.; Mellouk, A. Fog computing job scheduling optimization based on bees swarm. Enterp. Inf. Syst. 2017, 12, 373–397.

[25] Binh, H.T.T.; Anh, T.T.; Son, D.B.; Duc, P.A.; Nguyen, B.M. An Evolutionary Algorithm for Solving Task Scheduling Problem in Cloud–Fog Computing Environment. In Proceedings of the SOICT 9th Symposium on Information and Communication Technology, Da Nang City, Vietnam, 6–7 December 2018; pp. 397–404

[26] Jamil MA, Nour MK, Alotaibi SS, Hussain MJ, Hussaini SM, Naseer A. Software Product Line Maintenance Using Multi-Objective Optimization Techniques. Applied Sciences. 2023 Aug 6;13(15):9010.

[27] Jamil MA, Nour MK, Alotaibi SS, Hussain MJ, Hussaini SM, Naseer A. Adaptive Test Suits Generation for Self-Adaptive Systems Using SPEA2 Algorithm. Applied Sciences. 2023 Oct 15;13(20):11324.