Minimizing The Delay and Energy Consumption in Cloud and Fog Hybrid Environments Based on The Timing of Requests in The IOT Using Plant Defense Optimization Algorithm

Mostafa Rezaeizadeh Roukerd[1], Mehdi Jafari Shahbaz Zadeh[2*], Mahdiyeh Eslami[3]

1. PhD Student, Department of Electrical Engineering, Kerman Branch, Islamic Azad University,Kerman, Iran.
2. Assistant Professor, Department of Electrical and Electronics Engineering, Kerman Branch, Islamic Azad university , Kerman, Iran (Corresponding author).
3. Assistant Professor, Department of Electrical Engineering, Kerman Branch, Islamic Azad University , Kerman, Iran.

*** Corresponding author email address**: m-j-shahbazi@yahoo.com

Received: 2024-05-09 **Reviewed:** 2024-06-17 **Revised:** 2024-08-03 **Accepted:** 2024-08-16 **Published:** 2024-09-10

Abstract

This study addresses the interaction between IoT devices and their physical environment, highlighting its role in minimizing delays in IoT applications. The core challenge lies in scheduling IoT requests and efficiently allocating them to Fog and Cloud resources while ensuring fault tolerance. To tackle these issues, an intelligent optimization method based on the Plant Hormonal Defense Optimization (PDO) algorithm is proposed. The PDO algorithm, enhanced by applying penalties to restrict infeasible solutions, offers a high-quality solution within a reasonable computation time. The method's performance is evaluated using key metrics, such as average total delay in relation to data volumes, the number of lost requests, the delay difference between Fog and Cloud layers when altering processing speeds in the Fog layer, the 90th percentile delay as the number of servers in the Fog layer changes, and both Fog and Cloud breakpoints.The PDO algorithm is tested in a simulated environment that mimics real-world dynamics. Comparative analysis shows that the proposed method reduces delays by 23.84% to 48.51% when compared to other algorithms, demonstrating its superior performance.

Keywords: *IoT, Optimization, Plant Defense Optimization (PDO), Fog Computing, Cloud Computing.*

How to cite this article:

Rezaeizadeh Roukerd M , Jafari Shahbaz Zadeh M, Eslami M. (2024). Minimizing The Delay and Energy Consumption in Cloud and Fog Hybrid Environments Based on The Timing of Requests in The IOT Using Plant Defense Optimization Algorithm Management Strategies and Engineering Sciences, 6(3), 76-90.

1. Introduction

The Internet of Things (IoT), as an emerging technology, is poised to significantly impact our lives in the near future. This technology facilitates communication between various objects equipped with sensors and smart devices, allowing data and information collected by these objects to enhance daily life and provide better services to users. According to forecasts, the IoT is expected to grow exponentially in the coming years, driven by technological advances, increasing societal needs, and the vast potential it offers to both individuals and businesses. In 2014, the European Union launched research and innovation programs focused on IoT, setting specific goals to be achieved by 2020. This reflects the importance organizations place on IoT development and optimization, particularly in Europe, where there is a strong belief in the transformative potential of this technology [1]. Achieving IoT's full potential requires addressing critical challenges, such as optimizing request scheduling to reduce delays and energy consumption. Integrating cloud technologies, big data, and next-generation networks like 5G will also play a crucial role in realizing these goals, ultimately leading to improved services, resource efficiency, and the optimal use of IoT technology. IoT is both a concept and a paradigm that is increasingly present across various environments, encompassing both wireless and wired connections, and utilizing unique object addressing schemes. These features enable objects to interact and collaborate with other devices, facilitating the creation of new applications and services aimed at achieving common goals. However, there are numerous challenges in research and development as we strive to create an intelligent world. The convergence of real, digital, and virtual worlds is transforming domains such as energy, transportation, and smart cities, creating intelligent environments. The purpose of IoT is to allow objects to connect at any time, in any location, to any other object, entity, or human being. This connectivity, ideally achieved through any path, network, or service, defines the IoT [2-5]. A key distinction between fog computing and cloud computing is fog’s proximity to end users. In fog computing, services can be hosted on edge devices, such as access points, routers, switches, base stations, or even end-user devices. By bringing computation closer to the edge of the network, fog computing helps to address concerns about delays in cloud computing. The characteristics of fog computing are outlined in [6-8]. This research aims to model the problem of scheduling IoT requests from end devices and allocating them to appropriate resources in fog and cloud environments. We employ Integer Linear Programming (ILP) to model the minimization of service time and energy consumption for IoT requests. To achieve a high-quality practical solution within reasonable computational time, we propose a heuristic approach based on the Plant Hormonal Defense Optimization (PDO) algorithm. Plants use various defense mechanisms, including chemical compounds. For instance, when plants are attacked by pests or subjected to stressors such as drought or microbial infection, they release volatile organic compounds that trigger physiological responses in neighboring plants. Valdez et al. introduced a new biologically-inspired optimization technique based on this plant defense system, using the predator-prey model proposed by Lotka and Volterra. When a plant detects the presence of an invasive species, it initiates a chain of chemical reactions, releasing byproducts into the atmosphere that attract natural predators. In this paper, we propose a hormonal-based algorithm for optimization. The PDO algorithm aims to minimize delay and power consumption while ensuring that infeasible solutions are eliminated early in the process. This reduces the likelihood of selecting impossible solutions when generating new ones. The performance of the PDO algorithm is evaluated using a network simulator to assess its impact on delay and energy consumption [9-12]. Additionally, the proposed method is validated through the optimization of several mathematical functions, and its performance is statistically compared with other meta-heuristic optimization techniques.

The research method employed in this study is practical. It is based on the results of fundamental research, aimed at enhancing and refining behaviors, methods, tools, products, structures, and patterns used by human societies. The goal of applied research is to develop practical knowledge within a specific field.

2. Research Literature

2.1. Defining the Internet of Things

The Internet of Things (IoT) is not a single technology but a concept in which numerous objects are connected and activated. For example, streetlights connected in a network or objects embedded with sensors, image recognition, augmented reality, and near-field communication can make decisions related to positioning, resource management, and new services. These advancements create numerous business opportunities while also increasing the complexity of information technology. Fields such as distribution,

transportation, logistics, reverse logistics, and service environments are becoming more interconnected as information and "objects" interact, leading to the creation of new business processes or significantly more efficient and profitable entities.

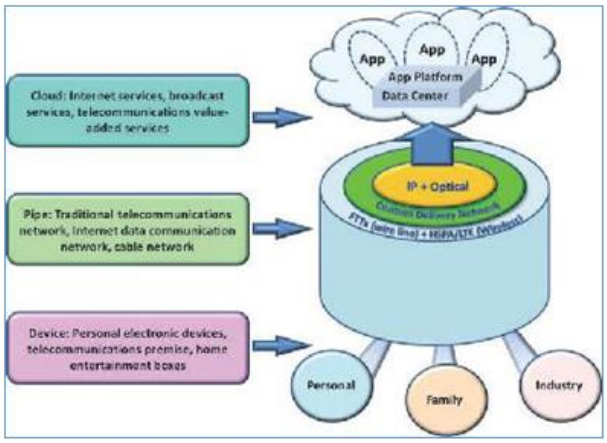

Figure 1. Factors going toward integrating and changing cloud, pipe, and technologies of the device [1]

The Internet of Things (IoT) offers a solution-based integration of information technology, encompassing both hardware and software used for data storage, retrieval, and processing, as well as communication technologies like electronic systems that facilitate interaction between individuals or groups. The rapid convergence of IT and ICT is taking place across three layers of technological innovation: cloud, data, and communication infrastructures (including pipes, networks, and devices), as shown in Figure 1 [1].

2.2. Security in IoT

Several advancements are necessary to establish a secure Internet of Things. Among the key concerns are: DDOS/DOS Attacks: While these attacks are well understood in the context of the current Internet, the IoT is also susceptible. Specific technologies and mechanisms must be implemented to protect essential infrastructures such as transportation, energy, and city systems from being disabled or compromised.

Threat Detection and Resilience: Robust systems are required to detect and mitigate specific IoT threats, including compromised nodes, malicious code, and hack attacks. Recovery strategies and resilience measures should be developed to counteract these threats effectively. Tool and Technique Awareness: Tools and techniques to monitor, manage, and secure IoT-based infrastructures need to be developed. These improvements should empower operators to protect IoT systems throughout their lifecycle and assist in adopting the most appropriate security measures during attacks. Access Controls: The IoT necessitates a range of access control mechanisms and licensing schemes tailored to support diverse user models. The heterogeneity of devices and gateways demands new lightweight access control designs. Autonomous Operation: The IoT must be capable of operating independently without human intervention. To achieve this, new technologies like machine learning will be crucial, enabling the development of a self-managed Internet of "smart" objects [3].

2.3. Computational Fog Environment

Cisco has introduced the concept of fog computing, which represents an extension of cloud computing that pushes services towards the edge of the network. In this model, distributed and virtualized environments can be efficiently managed to provide computing and network services between sensors and cloud data centers. While cloud computing offers data storage, computing power, and software services to end users, fog computing provides several advantages, including proximity to users, geographic distribution, and support for mobility. A simple three-tier hierarchy is depicted in Figure 2, demonstrating these layers [13].

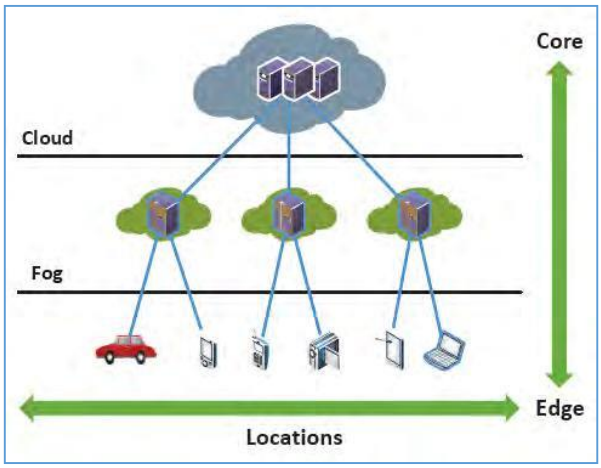

Figure 2. Three-Tier Hierarchy

In this framework, every intelligent device is connected to one of the fog machines. IoT environments consist of small, highly interconnected tools that operate through heterogeneous networks. The primary goal of such environments is to collect and process data from IoT devices to identify patterns, implement analyses, or optimize processes, ultimately enabling more intelligent and faster decision-making. The data in these environments can be classified into two categories:

- Small Data: Data transferred and permanently recorded from intelligent IoT devices.
- Large Flow or Big Data: Constant streams of data and knowledge stored and processed from centralized cloud storage [6].

2.4. Rsearch Background

Numerous studies have focused on task scheduling in fog environments, a few of which are highlighted here:

Bitam et al. proposed a bee-inspired algorithm to schedule tasks in fog computing environments. Their approach, based on the life of honeybees, utilized a linear combination of CPU runtime and memory allocation [14]. Al-Omari et al. explored error-tolerant models in IoT fog computing environments, proposing two methods: one without replication and one with minimal reproduction. In the non-replication model, other nodes in the fog environment replace the faulty node. Their model was analyzed in terms of energy consumption and running time [15]. Mohamed et al. studied error resilience in fog computing for smart city applications, emphasizing the reliability and robustness of IoT-based fog environments. Their research aimed to create a more reliable fog computing environment for smart city applications [16]. Yin et al. investigated task scheduling and resource allocation in fog computing for intelligent manufacturing. The study proposed a scheduling algorithm to ensure task completion and simultaneous processing, introducing a resource reallocation mechanism to reduce processing delays. The results demonstrated a significant reduction in execution delays and improved task concurrency in fog nodes [17]. Nguyen et al. introduced evolutionary algorithms for optimizing task scheduling in cloud-fog computing environments. The architecture of fog computing aims to complement cloud computing to meet IoT demands. The study employed an improved Particle Swarm Optimization (PSO) algorithm, tested on 11 datasets of varying sizes, with experimental results showing a 15.11% performance improvement [18].

3. Methodology

3.1. Proposed Method Framework

This paper proposes a framework consisting of two main components: (1) an innovative algorithm based on Plant Defense Optimization (PDO) to minimize request delays, and (2) an error tolerance optimization mechanism for handling request responses. To address the challenge of minimizing request delays, we propose an approach inspired by plant defense mechanisms. In this model, a mobile or resident user sends a service request to a fog node located at the network's edge. The node then transmits the request data and parameters as tasks to a manager node (usually located farther from the user). The manager node divides the request into smaller tasks and runs the PDO algorithm to determine the most optimized scheduling and allocation based on an objective function. To optimize energy management in fog

nodes, we combine runtime and energy consumption metrics. The manager node has access to all fog nodes and can distribute tasks to them. Finally, each fog node sends the results back to the manager node.

The following assumptions are made in this study:

Fog Node Concept: Each node refers to a fog computing node.

Memory Constraints: Each node has memory limitations.

Task Assignment: Multiple tasks can be assigned to a single node.

Energy Consumption: Energy consumption per node is calculated based on an established energy consumption model [18].

3.2. *Hormonal defense algorithm*

Throughout evolution, plants have developed various mechanisms to combat stress, including microbial infections. One such mechanism involves activating signaling pathways that lead to the expression of defense-related genes.

Historically, classic hormones such as ABA, SA, ET, JA, GAs, auxins, BRs, and CKs have been studied mainly for their roles in plant growth and development. However, recent studies have highlighted their importance in plant interactions and pathogenesis. During a pathogen attack, plants rapidly reallocate cellular resources from growth processes to defense mechanisms. Some of the key hormones involved are:

SA Hormone: The phenolic plant hormone salicylic acid (SA) is known for its role in thermogenesis, flowering, defense signaling, and systemic acquired resistance (SAR) [19]. Different plant species vary significantly in their endogenous levels of SA and responses to the hormone. For example, in tobacco (Nicotiana tabacum) and Arabidopsis, baseline SA levels are low (approximately 50 ng/g), but they can double during pathogenic infection. Genes related to the SA hormone are categorized into three groups, with new genes identified based on these relationships.

(1)

$$\vec{G}_n = \begin{cases} \vec{G}_1^{'SA} + w_1 \times \vec{r}_1 \times (\vec{G}_1^{'SA} - \vec{G}_2^{'SA}) & \alpha > rand \\ \vec{G}_1^{''SA} + w_2 \times \vec{r}_2 \times (\vec{G}_1^{'SA} - \vec{G}_2^{''SA}) & \alpha < rand < \beta \\ \vec{G}_1^{''SA} + w_3 \times \vec{r}_3 \times (\vec{G}_1^{''SA} - \vec{G}_2^{''SA}) & \beta < rand \end{cases}$$

The parameters α, β, and the condition β<α are used to determine the genes related to the SA hormone. In this context, α and β represent key thresholds that influence gene expression levels. The weight factor (W) is applied to adjust the influence of different genes, while r represents random numbers used to introduce variability and simulate the stochastic nature of biological processes. These parameters collectively help model the complex dynamics of gene regulation in response to the hormone SA.

3.3. *JA A Central Node In Plant Defense Signaling Network*

JA and its metabolites, collectively known as jasmonates (JAs), are important fat-derived regulators that play a fundamental role in defending plant growth processes [12].

One of the best examples to consider in the field of defense-related signal interference is the interaction between the SA and JA response paths [11]. Although research suggests that SA and JA routes work in contrasting ways, positive interactions have also been reported. For example, biosynthesis of JA in mutated plants is accompanied by a reduction in hydroperoxide lyase from OsHPL3 at the same time as an increase in the levels of SA. To simulate this approach in plants, we first select several genes from the SA family. These genes are combined with a weight less than JA genes (w5>w4) and then the new genes replace the previous genes and the relationship (2) is defined. This action applies to a small number of SA populations.

(2)

$$\vec{G}_n = w_5 \vec{G}^{JA} + w_4 \times \vec{r} \times (\vec{G}^{SA} - \vec{G}^{JA})$$

3.4. *ET Classic Defense Hormone*

ET is one of three classic defense hormones and is one of the key components of the hormonal composition released in the pathogen attack. Although there are exceptions, it is widely accepted that ET works with JA to strengthen immunity against necrotroph pathogens. Hormonal measurements showed that the ET pathway is active in sensitive plants but not active in resistant plants [9]. Thus, members of the ET family are combined with the JA family (relationship 3) as well as members of the JA family are merged with the ET family (relation 3).

(3)

$$\vec{G}_n^{JA} = \vec{G}^{JA} + w \times \vec{r} \times (\vec{G}^{JA} - \vec{G}^{ET})$$

(4)

$$\vec{G}_n^{ET} = \vec{G}^{ET} + w \times \vec{r} \times (\vec{G}^{ET} - \vec{G}^{JA})$$

3.5. *Hormone ABA*

Compared to SA, JA, and ET, the role of the ABA hormone in the plant's innate immunity is less well-known. Although the positive and negative effects of ABA on disease resistance have been reported, ABA acts mainly as a negative regulator of immunity [10]. Infection is usually associated with widespread reprogramming of ABA genes and biosynthesis and suggests that these pathogens have changed the course of rice ABA to cause disease. Therefore, the following mutations occur on the number δ percent of the population:

(5)

$$\vec{G}_n = \vec{G} + w \times \vec{r}$$

On the other hand, the lines for the mitogen-activated protein kinase OsMPK6 have been silenced by ABA, which generates ET too much and further resistance to M. oryzae. This approach indicates that the number of ETs is increased by the following relation, and the relation (6) is used to generate ET.

(6)

$$\vec{G}_n = \vec{G}^{ET} + w_1 \times \vec{r}_1 \times (\vec{G}^{ET} - \vec{G}^{ET})$$

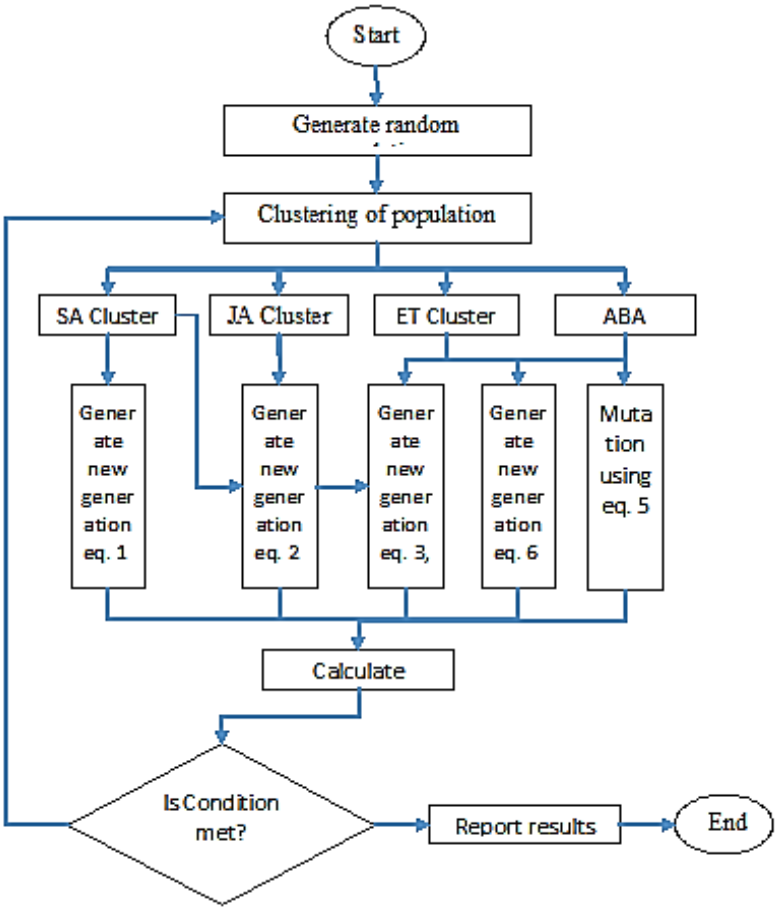

Figure 3. Flowchart PDO

3.6. *System and Work Model*

Table 1 provides an overview of the symbols used in the description of the algorithm, along with their corresponding definitions. The core approach of the proposed method is outlined as follows:

The proposed method aims to address the problem of minimizing both demand delay and energy consumption in cloud-fog computations. To achieve this, a new nature-inspired optimization approach, called the Plant Defense Optimization (PDO) algorithm, is introduced. This approach optimizes response delay time and energy cost to meet user demands efficiently. We evaluate the effectiveness of the proposed PDO-based optimization approach by comparing its performance to other existing methods, analyzing its efficiency in solving the demand-delay and energy consumption problems.

Table 1. The Signs Used Along with Their Definitions

Definition	sign	Definition	sign
Get results	T_R	Number of requests or tasks	N
Electric energy consumption per joule	EC(x) [J]	Request	R
Additional information sent to the Admin node	z	Task K from R's request	RK
Input Module Electrical Power	PI	Fog Mode	FN
Output Module Electrical Power	PO	Manage Mode	Admin
Electric power per watt for data storage	PS[W]	Average wait time in queue	q_t
Electric power consumption	ES	Processing time	CET
		Sent Time	T_S

4. Findings and Results

4.1. Evaluation Parameters

4.2. Evaluation Parameters for the Plant Defense Algorithm in Application Processing Optimization

These parameters focus on assessing the performance of the algorithm and its ability to reduce submission delays.

Average Total Delay vs. Data Volume:

This parameter measures the average total delay in responding to IoT requests relative to the total data volume transmitted. It is typically used to indicate how much the delay has been reduced as the data volume increases.

Number of Missed Requests vs. Average Request Size:

This parameter evaluates the number of requests lost due to network or server resource issues, in relation to the average request size. It shows how the Plant Defense Algorithm reduces the number of lost requests.

Ninety-May Delay Relative to Cloud:

This parameter measures the delay caused by the ninety-may node compared to the average delay caused by cloud nodes. It illustrates the performance difference between May nodes and cloud nodes in terms of delay reduction.

Fog-to-Cloud Delay by Changing Fog Layer Processing Speed:

This parameter compares the delays between fog and cloud under varying processing speeds in the fog layer.

Ninety-May Delay vs. Cloud by Changing Server Count in Fog Layer:

This parameter compares the delays between May nodes and cloud nodes by adjusting the number of servers in the fog layer.

Fog Breakpoints and Cloud Computing:

These parameters define the points where the Plant Defense Algorithm fails, resulting in the computation being transferred to the cloud.

Evaluation Parameters for the Plant Defense Algorithm in Application Fault Tolerance

The following parameters are used to assess the fault tolerance of requests within the Plant Defense Algorithm:

Guarantee Ratio (GR):

The ratio of tasks guaranteed to be completed.

Accepted Task Ratio (RTA):

The ratio of tasks accepted by the system for processing.

Number of Active Fog Nodes (NAF):

The total number of active fog nodes available for task processing.

Active Fog Time (FAT):

The duration for which fog nodes remain active during task execution.

Task Execution Time Ratio over Active Fog Nodes (RTF):

The ratio of task execution time distributed across active fog nodes.

Degree of Non-Equilibrium Loading Tasks (DIB):

A measure of how unbalanced the task distribution is across fog nodes.

4.3. Evaluation of the proposed algorithm for minimizing the delay of requests

The experiment consists of a set of 16 servers with an average processing speed of 500 packs per second and widely distributed from 50 to 1000 packs per second. The average delay of servers is set to 5 milliseconds per package. It is also distributed from 1 millisecond to 9.7 milliseconds. A total of 100 requests were used in the test. Priorities are distributed uniformly from 1 to 16. The runtime requirement is set to an average of 400 seconds with a 50-hour variance. Initially, the average size of the data starts from a small value, such that it reduces the deadline for requests. No request misses its execution deadline. Then, the average data size increases to see its effect on total delay and number of missed requests. Figure 3 and Figure 4 show the average

total delay and the number of requests lost against the average size of requests. PDO shows less total delay than other algorithms. The results of WFQ and PSQ are very close to each other because the allocation of requests in the resources is achieved based on the priority of both algorithms, however, the submission is different. RR represents the highest latency time, due to RR procedure, without consideration of request priorities. It can also be observed that PDO holds the record with no missing requests for longer than other algorithms. However, at an average of 6,500 packets, PDO cannot guarantee that all request deadlines are met because their service delays increase and their deadlines become critical. It is important to note that in all data, using PDO, can provide a practical scheduling solution in which all request deadlines must be met. However, simulation results show that it is not possible to overwrite all requests. This is shown in Figure 3 between 6000 and 8000, at average data size. After 8,000 packages, it becomes impossible to find the answer, and therefore, it is impossible to find a practical scheduling program. This is shown in the simulation because the actual delay of each package can be different from the average delay.

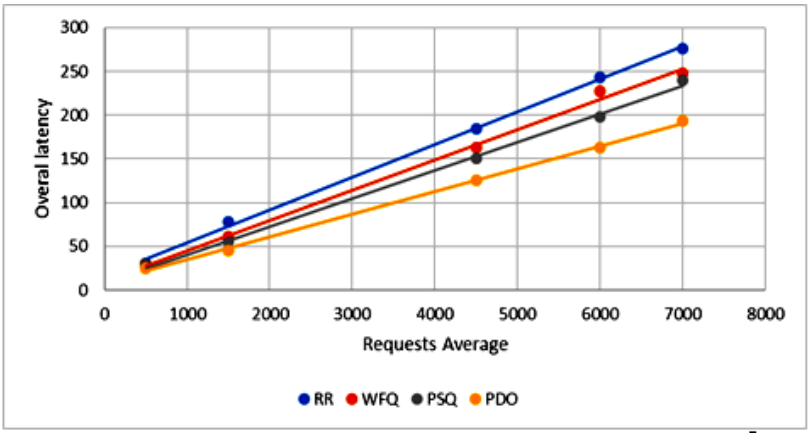

Figure 4. Average total delay against data volume

Figure 5. The number of lost requests versus the average size of the requests

4.4. Dynamic programming

In another experiment, 16 servers and 500 requests were used to assess the delay in processing applications. The requests were generated based on a Poisson distribution with an average input rate of one request per second. Priorities were assigned uniformly between 1 and 16. The average deadline for requests was set at 200 seconds, with a variance of 50 seconds. Resources were scheduled at 10-second intervals, and the average request size ranged from 1,000 to 10,000 packets. Figure 5 and Figure 6 depict the overall average delay and the number of lost requests in relation to the average data size. The Plant Defense Optimization (PDO) algorithm demonstrated the best overall delay performance compared to other algorithms. The results of WFQ and PSQ were very close to each other, but the

performance gap widened as the average data size increased. Figure 5 also shows that when the average data size is below 5,000 packets, PDO is able to process nearly all requests. At an average data size of 6,000 packets, PDO experienced no lost requests, while WFQ and PSQ lost approximately 10% and 6% of requests, respectively. The RR algorithm also exhibited a similar loss pattern. Beyond this point, as requests become more critical, the number of unprocessed requests rises. In the case of the PDO algorithm, lost requests begin to appear once the data size exceeds 6,000 packets. However, even in this scenario, PDO still outperforms the other three algorithms in terms of minimizing lost requests.

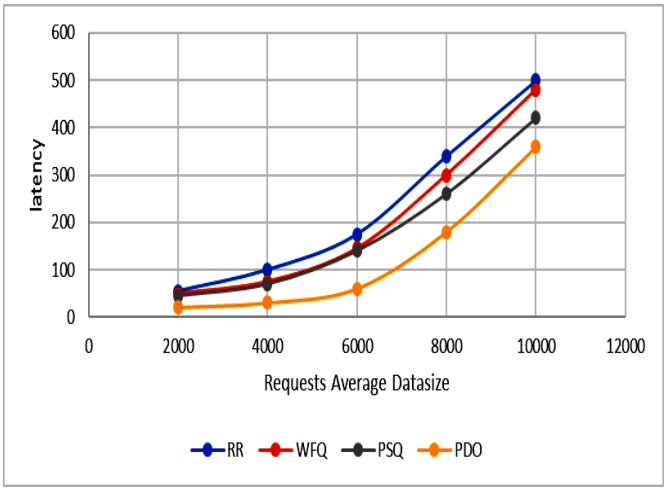

Figure 6. General average delay versus data volume

4.5. Comparison of cloud and hybrid architectures

The purpose of this study is to assess the service delays associated with resources that possess both cloud computing and fog computing features. Generally, cloud resources are classified as powerful due to their high processing capabilities, but they are also associated with higher average network transmission times and latency. On the other hand, fog resources, while having more limited processing power, are located closer to the network edge and therefore offer lower average delays. An experiment was designed to evaluate whether it is more efficient to use powerful cloud resources or to embed lower-power resources within the fog layer. To optimize response times and minimize delays in handling requests, the formulated model considers three key parameters:

- Average delay ratio, $\delta f/\delta c$
- Processing speed ratio, $P f/Pc$
- Resource Count Ratio, Nf/Nc

The effect of each parameter on service delay is studied independently by fixing two of them and changing only one. The delay obtained by changing these parameters is evaluated against a system that has a Cloud feature. This cloud-based system is home to a set of 4 high-performance, 5,000-pack/second-line servers. However, the average delay of these servers to 10 milliseconds per packet. To evaluate the delay, a total of 500 applications are used in this set of experiments. Their arrival follows the Poisson distribution with an average arrival time of 1 second. The delay is studied in comparison with the average size of the data, which varies from 1000 packages to 10,000 packages.

4.6. Effect of delay mean ratio

In this experiment, the number of Meme servers is four times that of cloud Nf /Nc = 4. Their processing power is only 10% cloud, Pf/Pc = 10%. The average delay ratio, $\delta f/\delta c$, varies from 1%, 10%, 20%, 50%, and 85%. Figure 7 shows the delay results and, as it is observed, an increase in the average delay ratio from 1% to 85% increases the delay until it reaches a point that passes through the fog lag. This indicates that the fog servers are far from edge devices and are closer to the fog servers.

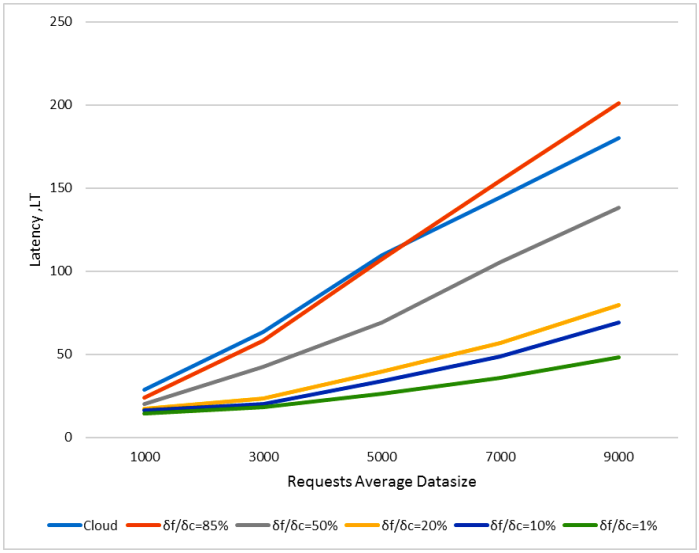

Figure 7. Ninety-Me delay to the cloud

4.7. The effect of the processing speed ratio

In this experiment, the number of fog servers is four times the number of cloud servers, $Nf/Nc = 4$. Their average delay is $\delta f/\delta c$ 10%. The processing power, Pf/Pc, varies to 3, 5, 7, 10 and 20 percent. Figure 8 shows delay results. As the results presented in Figure 8 show, reducing the speed ratio of processing from 20% to 3% increases the fog latency until it reaches a point that crosses the cloud latency. This is caused by low-process haze servers. In this case, even if these resources are closer to the edge than the cloud resources, the fog will offer high delays due to its low processing capabilities.

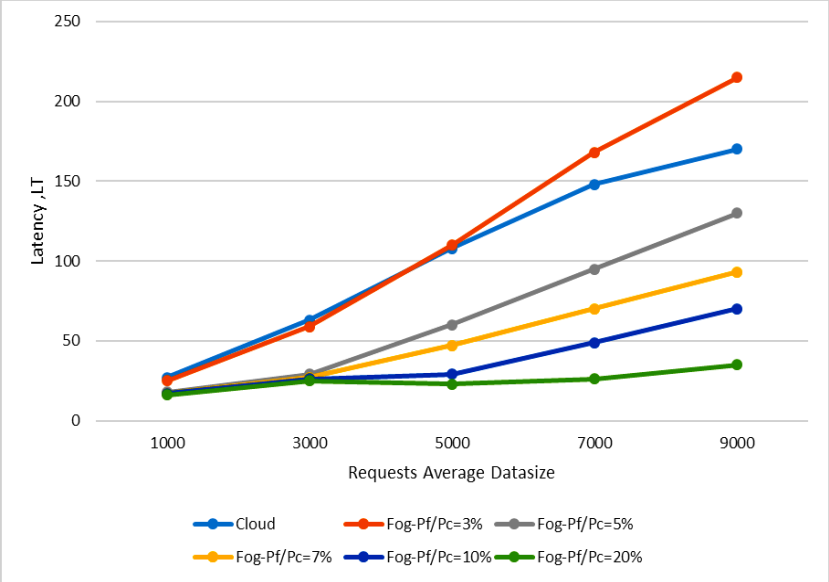

Figure 8. Fog-to-cloud delay by changing the processing speed in the fog layer

4.8. The effect of the ratio of the number of sources

In this experiment, The average delay of the May servers, is set to 10 percent of the mean cloud latency. The processing power of May servers is set to 10 percent of the features of cloud servers. The number of fog resources varies from cloud to 100%, 150%, 200%, 300%, 400%, 600% and 800%. Figure 9 shows the latency results for this experiment. As is evident from the test given in Figure 9, a decrease in the ratio of resources from 800% to 100% increases the fog delay until it reaches a point that passes through the cloud lag. This is because many foggy servers have very low resources.

Even if these resources are closer to the edge and have good processing capability, having fewer resources will adversely affect the delay.

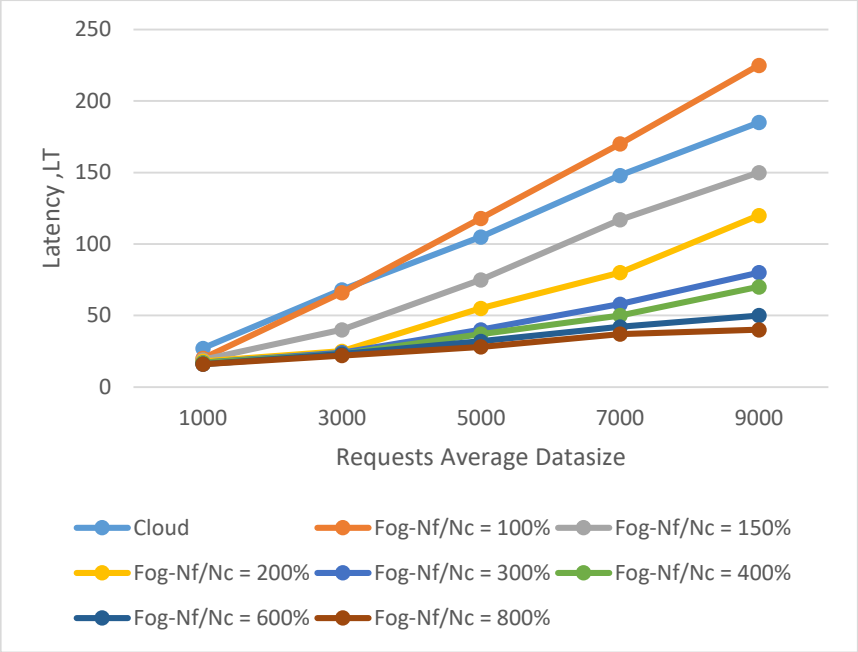

Figure 9. Until recently ninety-May compared to the cloud by changing the number of servers in the fog layer

Another experiment was conducted to identify failure points based on three parameters: average delay, processing power, and number of resources. The average data size was set to 5,000 packets. Figure 10 illustrates the experimental results, showing the confluence point between fog delays and cloud delays. The testing reveals that, under certain conditions, a set of fog servers provides better latency performance than a set of cloud servers. For example, for a given average delay ratio and processing power ratio between fog and cloud servers, the graph shows the number of servers at which fog performs better than cloud, and vice versa. The three most recent experiments demonstrated that fog computations encounter a delay threshold due to high delays (Figure 7), limited processing power (Figure 8), and a low number of fog servers (Figure 9), at which point fog's delay becomes greater than cloud computing's delay.

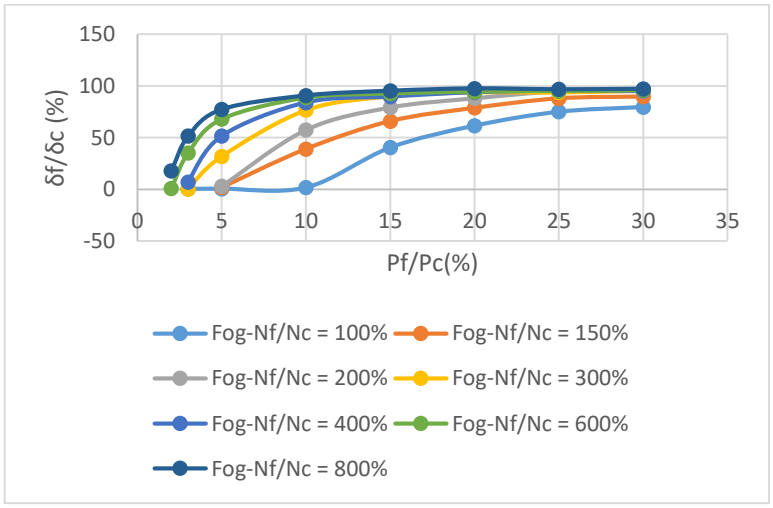

Figure 10. Foggy Breakpoints and Cloud Computing

Figure 11 represents the change in values of K (the weight of each part of the objective function). The form of 11-a shows the delay rate changes by changing its weight in the objective function, as it is shown, by increasing its weight, the value of the function F1 (the delay rate) also decreases. This is quite clear, because the more weight increases, the more fine increases (the algorithm moves to one-dimensional), so it is expected to perform better for one part of the function. Figure 11 shows the same behavior for the second part of the function: energy consumption. Here by increasing the weight of k2, the value of the second part of the objective function is decreased and then it is fixed.

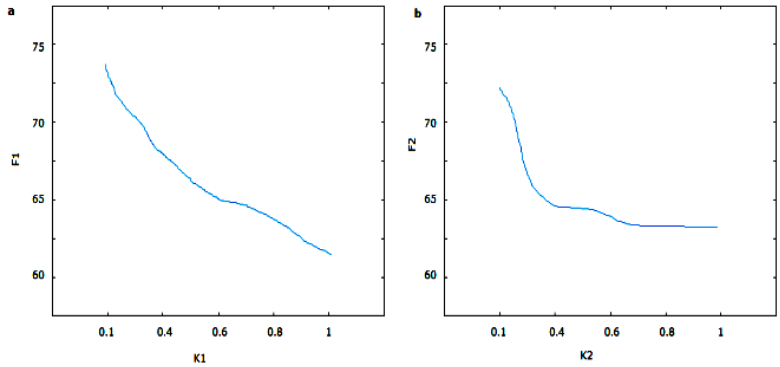

Figure 11. Changing K values on each part of the objective function

4.9. Evaluation of proposed algorithm for application fault tolerance

The experimental results in Table 2 show that the four algorithms of GRs are appropriate, but the DFTLA shows the best performance. The proportion of GR in NFTULA is relatively small. As mentioned, in NFTULA, the fault tolerance approach is not considered, so this is why the a decrease in GR in this algorithm compared to other methods. Because some tasks that can be performed successfully may be rejected by the system. For other algorithms, because they somehow take an error tolerance approach, GRs have been partially improved.

Table 2. Evaluation Parameters

NO-FT	SJD	PDO	Alg Metr.
0.76	0.95	0.99	GR
0.51	0.81	0.88	RTF
15.55	8.91	7.65	FAT ($\times 10^6$)
0.08	0.16	0.18	DIB

In the PDO approach, the Guarantee Ratio (GR) outperforms that of the SJD method, because in the PDO approach, not all nodes are used to store information. Instead, certain types of nodes have access to spine nodes, reducing energy consumption and ultimately improving the GR parameter. Energy consumption is also a key consideration, as the energy levels of fog nodes are limited and task scheduling must take this into account. In the PDO method, nodes are selected for processing that do not experience significant energy depletion during processing, thereby reducing the likelihood of task failure and increasing the GR. It is important to note that while the Resource Task Factor (RTF) parameter behaves similarly in both the PDO and SJD methods, the PDO method still produces better results overall. In contrast, the NO-FT method, which lacks an error control algorithm, performs significantly worse. Another evaluation parameter, the Fog Active Time (FAT), indicates the duration during which fog nodes are active. Since this parameter is closely related to the RTF, it is expected to yield similar results, as demonstrated in Table 2. The behavior of different methods regarding the FAT parameter is similar to their behavior with the RTF parameter. The final parameter in Table 2, the Degree of Imbalance (DIB), reflects the load distribution across the network. As shown, the load imbalance across various methods is relatively similar, except in cases where fault-tolerant methods are not used. This is because, in most methods, tasks are chosen to ensure a balanced load distribution across the network. As the task request rate (sent by the user) fluctuates, the number of active fog nodes

(NAF) also changes. Figure 12 illustrates the NAF variations over time for different algorithms in environments of various sizes—small, medium, and large. The number of active fog nodes in each algorithm is closely tied to the number of tasks being processed. As shown in Figure 12, when a large number of tasks are present in the environment, the PDO algorithm increases the number of active nodes to meet task scheduling demands.

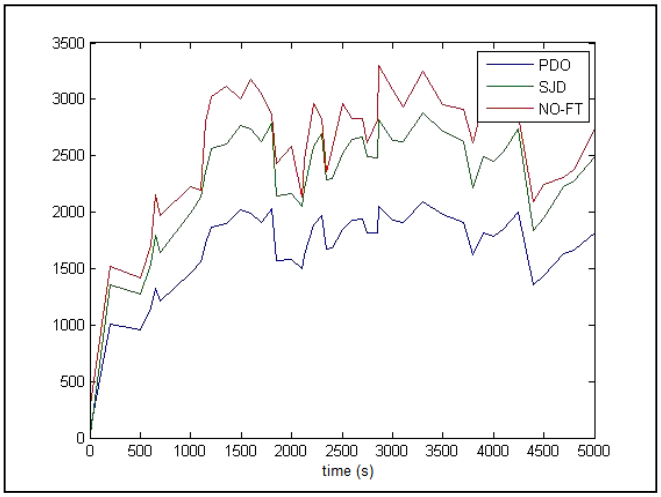

Figure 12. NAF Distribution Over Time

Another factor that affects NAF is a system error because if the node selected to process correctly processes the task, it does not require the choice of another ninety processors (activating another node). As mentioned above, in the PDO method, the load codes are selected which other nodes have access to. Therefore, data is available if it requires data. In the first 1000 seconds, algorithms in different environments have different behaviors, because initially the energies of noise are not reduced, fewer errors occur, and fewer jobs are posted. For example, in Figure 12, algorithms have performed better than SJD in terms of the NAF parameter in less than 1000 seconds, and over time this process is continued.RTA changes are shown over time in Figure 12. As shown in Figure 13, the RTA of the two other methods varies greatly in the first 1000 seconds and does not have uniform behavior, but slowly increases over time, eventually reaching GR. This means that when there are fewer active nodes in the environment, the cloud is not able to accept new tasks, but by increasing the active nodes, most accepted tasks will eventually be completed on time. On the other hand, at the beginning of the learning process of learning automata, since the automata have non-existent initial weight, the choice of ninety processors is merely random and the environment shows unexpected behaviors, but with time the automata shows more uniform behavior. However, since the model has been trained from the beginning in the PDO method, it has been better than the SJD method.

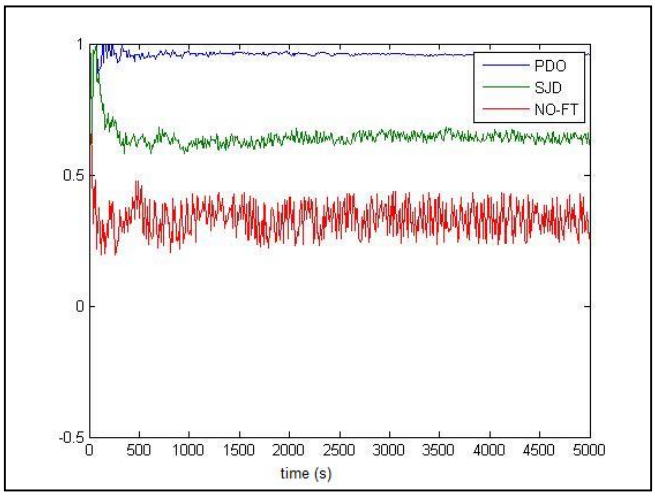

Figure 13. RTA Distribution Over Time

5. Conclusion

Among the various methods of modeling and optimization, the proposed algorithm based on Plant Hormonal Defense Optimization (PDO) has demonstrated significant improvements in performance. This approach utilizes the PDO algorithm, along with constraints that prevent impossible solutions, delivering high-quality, practical solutions within a reasonable computational time. To evaluate the performance of this algorithm, a set of evaluation parameters is used. These parameters include: average total delay relative to data volume, the number of lost requests, the delay of May nodes compared to cloud nodes, latency in May nodes relative to cloud nodes when processing speed changes in the fog layer, May node delay in comparison to cloud nodes by altering the number of servers in the May layer, and the breakpoints between May and Cloud. The results show that the PDO algorithm significantly reduces delays and energy consumption compared to other methods, making it a more efficient solution for managing IoT requests. Additionally, the PDO algorithm outperforms other approaches in fault tolerance evaluation metrics, such as Guarantee Ratio (GR) and Accepted Task Ratio (RTA). This method enhances the efficiency and utility of IoT, promoting better communication and the introduction of new services. Ultimately, the Internet of Things (IoT) offers a wide range of possibilities for improving everyday life. Its effects are substantial, from reducing energy consumption and improving healthcare services to increasing the efficiency of smart cities. These advancements demand further research in optimization and modeling to fully leverage this technology. The PDO algorithm has shown its potential as a key tool in this field, and with further optimization, it can significantly enhance the performance of IoT systems. In this context, optimal resource allocation is crucial. By incorporating the PDO algorithm into this process, the resource allocation system can be improved, leading to increased efficiency and effectiveness of IoT systems. Moreover, it is vital to consider the adaptability of this algorithm to different circumstances and environments. This flexibility allows the PDO algorithm to be applied in a variety of scenarios, boosting its efficiency and applicability in the IoT domain. Looking ahead, future research on IoT and its related challenges must focus on developing and optimizing algorithms and methods for resource management and productivity. The PDO algorithm represents an important step in this direction, offering an efficient and intelligent approach to managing IoT requests. It can serve as a powerful tool for improving the performance and efficiency of the Internet of Things.

Authors' Contributions

Authors equally contributed to this article.

Acknowledgments

Authors thank all participants who participate in this study.

Declaration of Interest

The authors report no conflict of interest.

Funding

According to the authors, this article has no financial support.

Ethical Considerations

All procedures performed in this study were under the ethical standards.

References

[1] O. Vermesan and P. Friess, *Internet of things: converging technologies for smart environments and integrated ecosystems*. River Publishers, 2013.

[2] J. Wan and et al., "Fog Computing for Energy-aware Load Balancing and Scheduling in Smart Factory," *IEEE Transactions on Industrial Informatics*, 2018, doi: 10.1109/TII.2018.2818932.

[3] P. Zhang, M. Zhou, and G. Fortino, "Security and trust issues in Fog computing: A survey," *Future Generation Computer Systems*, vol. 88, pp. 16-27, 2018, doi: 10.1016/j.future.2017.09.002.

[4] A. A. Mutlag and et al., "Enabling technologies for fog computing in healthcare IoT systems," *Future Generation Computer Systems*, vol. 90, pp. 62-78, 2019, doi: 10.1016/j.future.2018.07.049.

[5] Y. Pan and G. Luo, "Cloud Computing, Fog Computing, and Dew Computing," *ZTE COMMUNICATIONS*, vol. 15, no. 4, 2017, doi: 10.1109/MCC.2017.4250924.

[6] F. Bonomi and et al., "Fog computing and its role in the Internet of Things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, 2012, doi: 10.1145/2342509.2342513.

[7] O. Salman and et al., "IoT survey: An SDN and fog computing perspective," *Computer Networks*, vol. 143, pp. 221-246, 2018, doi: 10.1016/j.comnet.2018.07.020.

[8] J. Zhu and et al., "Improving website performance using edge servers in fog computing architecture," in *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*, 2013: IEEE. [Online]. Available: https://ieeexplore.ieee.org/document/6525539. [Online]. Available: https://ieeexplore.ieee.org/document/6525539

[9] D. De Vleesschauwer, G. Gheysen, and M. Höfte, "Hormone defense networking in rice: tales from a different world," *Trends in plant science*, vol. 18, no. 10, pp. 555-565, 2013, doi: 10.1016/j.tplants.2013.07.002.
[10] R. Garg, A. K. Tyagi, and M. Jain, "Microarray analysis reveals overlapping and specific transcriptional responses to different plant hormones in rice," *Plant signaling & behavior*, vol. 7, no. 8, pp. 951-956, 2012, doi: 10.4161/psb.20910.
[11] R. Li and et al., "OsNPR1 negatively regulates herbivore-induced JA and ethylene signaling and plant resistance to a chewing herbivore in rice," *Physiologia Plantarum*, vol. 147, no. 3, pp. 340-351, 2013, doi: 10.1111/j.1399-3054.2012.01666.x.
[12] X. Peng and et al., "Constitutive expression of rice WRKY30 gene increases the endogenous jasmonic acid accumulation, PR gene expression, and resistance to fungal pathogens in rice," *Planta*, vol. 236, no. 5, pp. 1485-1498, 2012, doi: 10.1007/s00425-012-1698-7.
[13] V. B. Souza and et al., "Towards distributed service allocation in fog-to-cloud (F2C) scenarios," in *Global Communications Conference (GLOBECOM), 2016 IEEE*, 2016: IEEE, doi: 10.1109/GLOCOM.2016.7842341.
[14] S. Bitam, S. Zeadally, and A. Mellouk, "Fog computing job scheduling optimization based on bees swarm," *Enterprise Information Systems*, vol. 12, no. 4, pp. 373-397, 2018, doi: 10.1080/17517575.2017.1304579.
[15] R. Al-Omari, A. K. Somani, and G. Manimaran, "Efficient overloading techniques for primary-backup scheduling in real-time systems," *Journal of Parallel and Distributed Computing*, vol. 64, no. 5, pp. 629-648, 2004, doi: 10.1016/j.jpdc.2004.03.015.
[16] N. Mohamed, J. Al-Jaroodi, and I. Jawhar, "Towards fault-tolerant fog computing for IoT-based smart city applications," in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, 2019: IEEE, doi: 10.1109/CCWC.2019.8666447.
[17] L. Yin, J. Luo, and H. Luo, "Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4712-4721, 2018, doi: 10.1109/TII.2018.2851241.
[18] B. M. Nguyen, H. Thi Thanh Binh, and B. Do Son, "Evolutionary Algorithms to Optimize Task Scheduling Problem for the IoT Based Bag-of-Tasks Application in Cloud-Fog Computing Environment," *Applied Sciences*, vol. 9, no. 9, p. 1730, 2019, doi: 10.3390/app9091730.
[19] A. C. Vlot, D. M. A. Dempsey, and D. F. Klessig, "Salicylic acid, a multifaceted hormone to combat disease," *Annual review of phytopathology*, vol. 47, pp. 177-206, 2009, doi: 10.1146/annurev.phyto.050908.135202.