Journal of Data Acquisition and Processing
ISSN: 1004-9037
https://sjcjycl.cn/
DOI: 10.5281/zenodo.98549859

OPTIMIZING SCHEDULING PROBLEM IOT BASED CLOUD COMPUTING ENVIRONMENT USING EAS (IBEA ALGORITHM)

Syed Mutiullah Hussaini
Part-time Research Scholar, Jamal Mohammed College, Baharathidasan University, Trichy, Tamil Nadu State, India

Dr.T.Abdul Razzak
Associate Prof. in Computer Science Dept., Jamal Mohammed College, Baharathidasan University, Trichy, Tamil Nadu State, India

Abstract - Recently, the infrastructure of cloud computing has gained most importance due to Internet of Things' (IoT) ongoing advances, which produce massive amounts of data. For satisfying the demands by the IoT device network. The proposed Fog computing system is expected to represent the upcoming stage of cloud-based computing. One of the difficulties fog computing faces is the proper allocation of computing power to decrease processing times and operational costs. In this research paper a novel method for reducing operational costs and optimizing task scheduling s in a cloud-fog environment is discussed. This paper proposes an IBEA algorithm to assign service requests with the objective to minimize completing time and operating cost.

Keywords--- Cloud computing, IoT things, IBEA algorithm, optimization, Quality of Service

1. Introduction

A significant advancement in information and communication technologies sector recently has been the Internet of Things (IoT). With IoT, Internet connectivity is extended to a wide range of objects and devices (devices, machineries, automobiles, etc.), in addition to conventional smart gadgets, such as handsets and laptops. To implement a variety of applications and services, including: energy management, vehicle networks, traffic control, health care, and medical treatment. This generates an enormous amount of data that must be processed, stored, and evaluated in order to produce useful data that will fulfill the objectives and goals of users. Additionally, the quantity and scope of apps and services are expanding quickly, necessitating processing power that even the most advanced smart devices cannot now provide. The cloud infrastructure is recognized as an important computing hub that enables dynamic resource sharing and distribution among users using virtualization technologies. It is an exciting opportunity to promote IoT enhancements. Limitations of present smart gadgets (battery life, processing speed, storage space, and networking bandwidth) can be decreased by shifting labor- and resource-intensive processes to a dependable computing platform like the Cloud and allowing basic tasks to be handled by smart gadgets.

But when IoT and cloud computing are combined, other issues appear. Information Handling Services (IHS) Arcade predicts that the number of installed IoT devices will increase from 15.4 billion in 2015 to 30.7 billion in 2020 and 75.4 billion in 2025. The traditional centralized processing elements of the Cloud architecture (where computing and storage resources are

pooled and kept in many data centers) would not be able to satisfy the needs of IoT applications given this sharp increase in the number of connected devices. The main cause is the significant distance between IoT devices and the cloud. Massive amounts of data being transmitted from Latency will result from IoT devices communicating to the cloud via the Internet, especially near barriers, as well as put a strain on network bandwidth and performance. 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, workstations, personal computers, etc.) now have an increasing number of processing, storage, and communication capabilities as a result of advancements in both hardware and software technology.

A novel concept of cloud computing called fog computing [1], first put forth by Cisco, can turn A networked interface into a distributed computing environment that can support Internet of Things applications. Fog computing's purpose is to bring capabilities for processing and storage closer to users by extending cloud computing to Internet of Things (IoT) devices that generate and consume data.

The Cloud-Fog computing architecture has many benefits, such as decreased latency, less network traffic, and better energy efficiency, but it also has certain drawbacks. Resource allocation and task scheduling are two of them. In the Cloud-Fog system, job scheduling is done to the advantage or service suppliers, respectively. Several criteria are causing user's concern, including timeline, price, deadline, security, and cost. On the other hand, load balancing, resource usage, and energy efficiency are the goals of the service providers. Response time is essential when determining the QoS because it has a direct impact on the user's experience. Users are also very concerned in the cost of implementation as another factor. A task schedule that cuts down on completion time and costs will fulfill the SLA (Service Level Agreement) reached with users.

We concentrate on the scheduler in this investigation. issue to address this issue, a Time-Cost aware Scheduling (TCaS) method is suggested. The TCaS technique's fundamental goal is to effectively manage implementation time with costs in order to complete 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 works are shown in the pertinent Unit 2. In the context of cloud-fog computing, the problem of job scheduling is represented mathematically described in Unit 3. Our suggested IBEA algorithm job scheduling problem is presented in full in Unit 4. Unit 5 presents Crossover operator. Unit 6 presents the experimental findings. Unit 7 brings the article to a close and discusses potential future efforts.

2. Related Works

2.1 IoT with Fog Computing

IoT applications are generating a number of challenges for the centralized Cloud architecture. IoT, for instance, cannot handle real-time and low latency applications like linked vehicles [1], intelligent traffic signals [5], etc. These problems can be resolved through fog computing. the distinctions between cloud computing and fog computing.

Fog computing, in the opinion of Chiang and Zhang [6], can aid in addressing issues with the Internet of Things.

.

Survey papers have been written about several facets of fog computing. Shi et al. [7], for instance, looked at the crucial components for healthcare applications of fog computing. In addition, Yi et al[8].'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 [9]. This offers a cloud-based interface to stop Fog colony colonies from acting on their own. Fog cells are instead analyzed, controlled, and monitored by cloud-based infrastructure. Additionally, Bonomi et al. [10] explored the integration of IoT with Fog computing by evaluating essential elements of Fog computing and how Fog complements and extends 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, assessing, and modeling delays in IoT-Fog-Cloud systems was put forth by Yousefpour et al. [11]. 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 reduce 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 et al. [12]. 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 et al. [13] developed a Modular Fog (MF) in an organizational computing framework that can deploy applications for the Internet of Things 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 et al. [14] on its difficulties and distinguishing characteristics. They highlighted the variations between Mobile cloud computing, Mobile edge computing, Cloud computing, Fog computing, and Edge computing. They looked at networking setups, different Fog computing metrics, and Fog node settings.

Job scheduling utilizing adjusted PSO calculation in distributed computing climate by Abdi et al. [21] fostered a modified particle swarm optimization (MPSO). They differentiated it to two notable heuristic strategies, specifically PSO and GA, 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 et al. [20] addressed load balancing in fog computing with a focus on enhancing user experience (QoE). First, they divided up available computer power into little cells. Following that, fog computing configures clusters based on a set optimization target for arrival time, delay, and other characteristics for each user's requests. With a simple computer architecture, they were able to produce good results, however, if the Fog computing system is greatly enlarged, the strategy may become challenging.

.

In a Cloud-Fog computing environment, Huynh Thi Thanh Binh [15] suggested a job scheduling technique based on Genetic Algorithms (GA). This approach seeks to achieve a favorable time/cost trade-off.

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. Consider the cloud system (CS), which is made up of Npm physical machines (PM) and Nvm virtual machines (VMs) on each PM [7].

3.2. Description of the Problem

Requests from BoJ applications are broken down into manageable, independent jobs, whenever the Cloud-Fog computing framework transmits them to the Fog layer for being handled. The number of instructions, the amount that is needed for memory, and the size of the input and output files are the characteristics of each job.

Assuming that Jk stands for the kth job , the system receives a collection of n distinct jobs each time, as follows:

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

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} K_{Fog}), which is expressed as

$K = \{K_1, K_2, K_3, ..., K_n\}$, ……(2)

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

Every Job J_k

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

$K_i Jobs = \{ J^i_x , J^i_y , ..., J^i_z\}$ …….. (3)

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

$NodeJobs = \{J^a_1 , J^b_2 , J^c_3 , ..., J^p_n\}$ ……..(4)

The execution time (EXT) required by a node K_i to perform all jobs assigned for a collection of jobs is: (objective -1)

where *ExeTime*(*Ti)*

$$EXT(K_i) = \sum_{J\,ik \in KiJobs} \quad ExeTime(J^i_k) = \frac{\sum J^i_k \in KiJob \;\; length(J_k)}{CPUrate(K_i)} \qquad (5)$$

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

$$ExeTime(J^i_k) = \frac{length(J^i_k)}{CPUrate(K_i)} \qquad \text{----------(6)}$$

, where length(J_k) is the amount of instructions in task J_k and CPU rate(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.

Make-span 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 *Make-span*.

$$Make\text{-}span = \max_{1 \le i \le m} [EXT(K_i)] \qquad (7)$$

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.

$MinimalMakespan = EXT(K_1) = ... = EXT(K_m)$

thus,

$$MinimalMakespan = \frac{\sum 1 \le k \le n \; length(Jk)}{\sum 1 \le i \le n \; CPUrate(Ki)} \qquad (8)$$

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 Ki processing task Jk is given as follows:

$$Cost(Jk\ i) = cp(\mathrm{J}k\ i) + cm(\mathrm{J}k\ i) + cb(\mathrm{J}k\ i) \quad (9)$$

Each cost is computed using Equation (5) as follows.
As determined by processing cost:

$$cp(J^i_k) = c1 * ExeTime(\mathrm{J}^i_k) \quad (10)$$

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

$$cm(J^i_k) = c2 * Mem(J^i_k) \quad (11)$$

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:

$$cb(J^i_k) = c3 * Bw(J^i_k) \quad (12)$$

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

$$TotalCost = \sum_{J^i_k \in NodeTasks} Cost(J^i_k) \quad (13)$$

When each work is given to the least expensive node, MinTotalCost—the least amount necessary to perform a set of jobs J—can be calculated. It is simple to discover which node completes job J_k at the lowest cost, or MinCost, given the information of each node (J_k). As a result, MinTotalCost is particular to a certain collection of jobs J and is defined as:

4. IBEA for Job Scheduling Problem

4.1 Population Initialization

Assuming 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 stochastically.
Fig. 1 IBEA

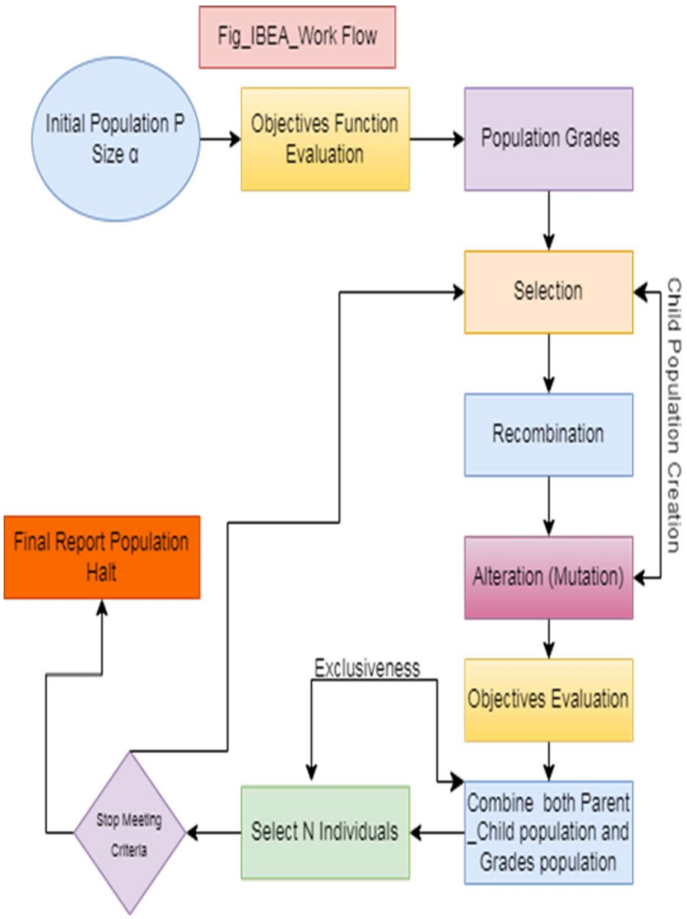


Figure 1: IBEA Workflow

IBEA_Step by Step Process_Algorithm

Input: ᵞ,N,β

1Describe preliminary *population X* of size **ᵞ**

2Set the unit counter *k* to 0

3 while *population size X* docs not exceed **ᵞ do**

4 **foreach** *Population Discrete* **do**

5 **Compute** *fitness* values of *Discrete*

6 **endforeach**

7 Select an discrete *x ε X* with the tiniest fitness value

8 Eradicate *x* from the *population X*

9 Update the *fitness* values of the remaining *discrete's*

10 **if** *k ≥ N or another ending norm is satisfied*

then

11 Consider *discrete's* not dominated in *X* as the conclusion path

12 **else**

13 Execute dual event collection with replacement on *X* in order to plug the provisional coupling pool *X'*

14 Mark re-combination and change operatives to the coupling pool X' and increase the resultant descendants to *X*

15 Raise the unit counter *k* and go to line **5**

16 **end if**

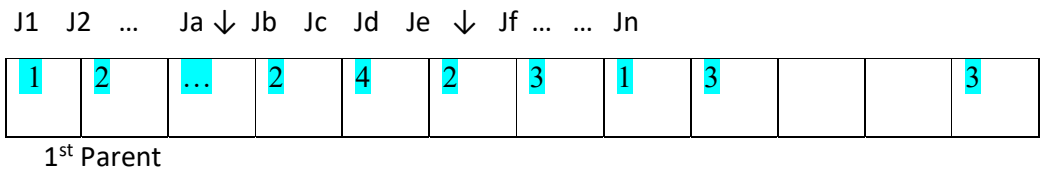
17 end while

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

Using chromosome encoding as an array of integers, the dual-point crossover process is used to generate youngsters that inherit advantageous genetic factors from parents. In Figure 2, this process is illustrated. A new person is produced by randomly selecting two crossover locations, where the first parent exchanges the middle gene segment with the second parent while leaving the other genes intact.

Parental selection affects the algorithm's effectiveness in the crossover process over the population. Everyone has a crossover frequency. The population's second parent is then chosen at random to participate 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 higher fitness values are more likely to be chosen as parents, ensuring that superior gene segments are more likely to be kept in the following generation.



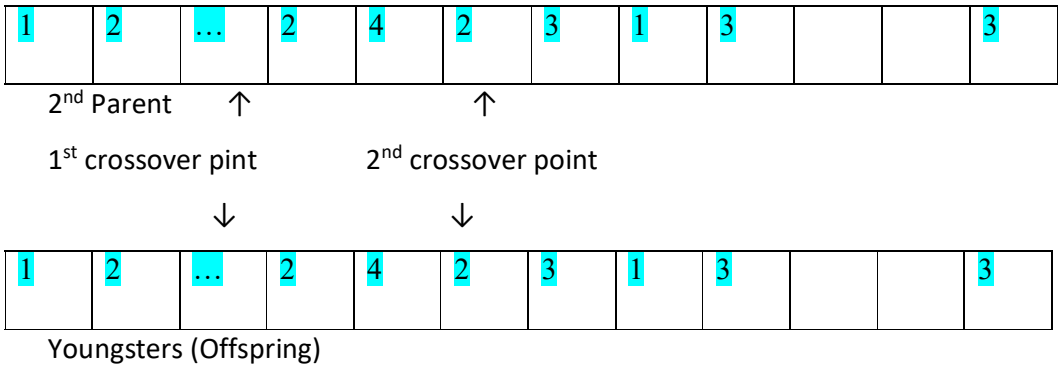


Figure 2: Chromosome Encoding

6. Performance Assessment

In this section, we present the assessment of the proposed IBEA algorithm based on the results data sets.

6.1 Experimental Settings & Results

Cloud and fog nodes have different resource use costs and processing parameters. We probably considered each node's processing rate with CPU, memory, and bandwidth consumption costs to be a representation of its individual processing capabilities. In the following Table 1 shows the characteristics of different processing nodes built the Cloud-Fog Structure. Fog layer nodes with minimal computing capacity include routers, gateways, workstations, and personal PCs. In the cloud layer, jobs are handled by servers or virtual machines in high performance data centers. As a result, Cloud nodes process information significantly more quickly than Fog nodes. The cost of consuming resources, however, is higher in the Cloud than in the Fog.

Table 1: Cloud-Fog Structure's characteristics.

Parameter(s)	**Fog Setting**
Number of Nodes	10
CPU rate	[400,1200]
CPU usage cost	[0.09,0.3]
Memory Usage Cost	[0.01,0.03]
Bandwidth usage cost	[0.09,0.02]

The Cloud-Fog structure is accountable for performing all requirements from the users. Every request is separated apart into a list of jobs, which are then reviewed and their expected amount of resources are determined. The number of jobs in the collection may be very different in size

depending on the workload of each request. As a result, 10 datasets containing between 40 and 500 jobs were made to take part in our experiment.

Table 2: IBEA Parameter settings

Parameter(s)	**Values**
Crossover rate	70%
Mutation rate	70%
Number of generations	500
Population size	200

Table 2 shows the best parameters setting for the proposed problem. Meanwhile our suggested IBEA algorithm is based on Genetic Algorithm.

6.2 Testing Results

Numerous tests were carried out with two separate objectives to evaluate our processes. In the first stage, a range of jobs were carried out locally in a network of linked Fog nodes; in the second, the Fog layer cooperated with a number of cloud nodes to process user requests.

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 connected to one another.

The IBEA algorithm's scheduling time and processing costs are shown in the following Table 3. Make-span and total costs were the two key factors that contributed to the function of fitness. As a result, the IBEA algorithm showed superiority on practically all datasets in terms of both time and cost.

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

Table 3 Make-span and total cost for IBEA

	Make-Span (s)	**Cost**
Number of Jobs	**IBEA**	**IBEA**
40	192.40*	735.58

80	395.55	1545.32
120	609.65	2365.47
160	822.67	3275.15
200	945.85	3835.35
250	1,285.95	4988.95

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

In this instance, studies were carried out on the entire Cloud-Fog system, which included 10 Fog nodes and 3 distant Cloud nodes. In contrast with objective 1, Fog nodes' attribute values were identical, and duties appeared to be divided among them evenly. 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. Conclusions and Next Steps

In the present study, we focused on the job scheduling challenge for IoT applications in the Cloud-Fog computing paradigm. How well the IBEA algorithm performs, which is based on an evolutionary process, was assessed with respect to two objectives minimization. The proposed IBEA algorithm generates the trade-off between time and cost execution, particularly in getting substantially lower scheduling duration also discussed in [15]. 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

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

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

Chiang, M.; Zhang, T. Fog and IoT: An overview of research opportunities. IEEE Internet Things J. 2016, 3, 854–864.
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.
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.
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.
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.
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.
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.
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.
Mahmud, R.; Kotagiri, R.; Buyya, R. Fog computing: A taxonomy, survey and future directions. In Internet of Everything; Springer: Singapore, 2018; pp. 103–130.
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.
He, J.; Cheng, P.; Shi, L.; Chen, J.; Sun, Y. Time synchronization in WSNs: A maximum-value-based consensus approach. IEEE Trans. Autom. Control 2014, 59, 660–675.
Li, D.; Sun, X. Nonlinear Integer Programming; Springer Science & Business Media: Berlin, Germany, 2006; Volume 84.
M. Abid Jamil and M. K. Nour, "Managing software testing technical debt using evolutionary algorithms," Computers, Materials & Continua, vol. 73, no.1, pp. 735–747, 2022.