IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Ingénierie des Systèmes d'Information
Vol. 26, No. 2, April, 2021, pp. 211-224
Journal homepage: http://iieta.org/journals/isi

A Survey on Various Methods and Algorithms of Scheduling in Fog Computing

Raouf Belmahdi[1], Djamila Mechta[1*], Saad Harous[2]

[1] LRSD Lab, Computer Science Department, College of Sciences, University of Sétif1, Sétif 19000, Algeria
[2] Dept. of Computer Science and Software Engineering, College of Information Technology, UAE University, Al Ain 15551, UAE

Corresponding Author Email: mechtadjamila@univ-setif.dz

https://doi.org/10.18280/isi.260208

Received: 4 January 2021
Accepted: 16 March 2021

Keywords:
cloud computing, Fog Computing, internet of things, scheduling algorithms, optimization methods, heuristic, meta-heuristic

ABSTRACT

The rapid deployment of IoT in different areas generates a massive amount of data transferred to the Cloud. To solve this challenge a new paradigm, called Fog Computing, is located at the edge of the network and close to the connected objects. Its main role is to extend the capacities of Cloud and improve the performance and the QoS required by the applications by the use of different methods and techniques based on scheduling algorithms. In this paper, we review various recent studies available in the literature that are interested in the scheduling methods and algorithms used in Fog computing. The use of fog layer, in solving optimization problem, is faced with serious challenges. Therefore, to help practitioners and researchers, we present an in-depth overview of Fog Computing studying various scheduling methods and algorithms. We analyze, compare and classify these different scheduling approaches according to the nature of the algorithm used in the scheduling, the QoS optimized by the proposed approach and the type of applications in order to show what is suitable for critical IoT (CIOT), massive IoT (MIOT) and Industry IoT (IIOT). Finally, we present a comparison of the different simulation tools used to evaluate these approaches to guide fog computing developers/researchers which tool is suitable and most flexible for simulating the application under consideration.

1. INTRODUCTION

The Internet of Things (IoT) is deployed in most areas [1, 2]. This technology is based on connected objects, which are widely used. Cisco estimates that there will be around 50 billion connected devices by 2020 [3]. They generate a massive amount of data that needs to be transmitted, stored, processed and analyzed in Cloud data centers. This increases the Cloud load and transmission latency; therefore, the Cloud cannot meet the QoS requirements of IoT critical applications such as latency sensitive applications. These challenges require a new architecture to deal with it [4].

1.1 Overview of Fog Computing

In 2012, Cisco proposed a new paradigm named Fog Computing [5]. It is a highly virtualized platform that provides computation, storage, and networking services between end devices and traditional Cloud Computing Data Centers [5]. Its main role is to extend the Cloud to be closer to the objects that produce and act on IoT data [6]. The Fog Computing is defined by OpenFog Consortium [7] as "*horizontal, system-level architecture that distributes computing, storage, control and networking functions closer to the users along a cloud-to-thing continuum*".

1.1.1 Architecture of Fog Computing

Recently several research studies proposed a Fog Computing architecture [7-10]. This architecture is divided into several layers, the number of layers depends on the point of view of the author or the application [8]. The majority of these studies presented a three hierarchical layers architecture [9, 11-14] as shown in Figure 1.

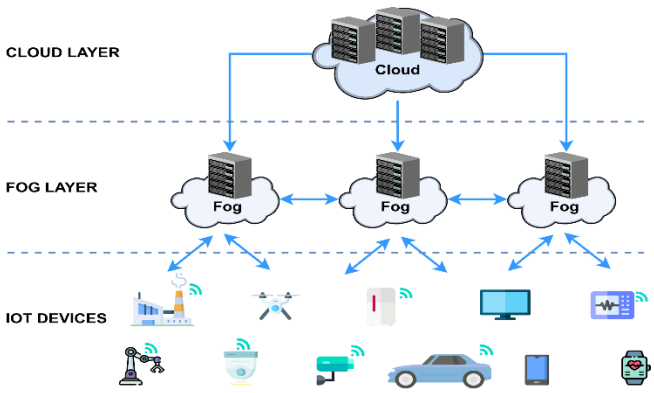


Figure 1. A hierarchical architecture of Fog Computing

1.1.2 Characteristics of Fog Computing

In the literature, several researchers [8, 13, 15-17] have proposed a comparison between the characteristics of Cloud and Fog Computing. Based on these studies, it is noticed that the Cloud Computing represents a large data centers available over the Internet, geographically centralized, on the other hand, the Fog Computing constituted of several small data centers, geographically widely distributed. This provides services to user requests at the edge of networks where the distance between client and server is one or few hops, which allows location awareness and reduces latency. In terms of resources,

Cloud Computing has a high resources capacity (CPU, RAM, Storage, Bandwidth), which requires high energy consumption, and high-cost resources' usage. By comparison, Fog Computing, has less resource capacity at lower cost, and which reduces energy consumption linked to the small data center, located at the edge networks.

The common differences between the characteristics of these two paradigms is summarized in Table 1.

Table 1. A comparison between the characteristics of Cloud and Fog Computing

Characteristics	Cloud Computing	Fog Computing
Latency	High	Low
Delay Jitter	High	Low
Realtime application handling	Difficult	Achievable
Mobility	Limited	Supported
Geographical distribution	Centralized	Distributed
Server nodes' location	Within the Internet	At the edge of the network
Distance between client and server	Multiple hops	Single/Few hops
Location awareness	No	Yes
Number of server nodes	Few	Large
Computation capability	High	Low
Storage capacity	High	Low
Energy consumption	High	Low
Power source	Direct power	Direct power, Battery, Green energy
Computation cost	High	Low
Bandwidth cost	High	Low

1.1.3 Advantages, challenges and issues of Fog Computing

According to the comparative table (Table 1), we observe that the Fog Computing paradigm offers several advantages to applications such as: reduction of the computational load and the use of resources of Cloud servers [16], reduces network traffic, suitable for IoT applications [10], ensures low latency and real-time interactions, and supports for mobility [9].

Despite the advantages offered by Fog Computing, it poses many challenges and issues: security and privacy, energy management [9, 16], resource management and scheduling [8].

1.2 Scheduling in Fog Computing

The scheduling is a NP-hard problem, as defined by ***M.Pinedo*** in [18]: "*Scheduling is a decision-making process that is used on a regular basis in many manufacturing and services industries. It deals with the allocation of resources to tasks over given time periods and its goal is to optimize one or more objectives*". Scheduling presents one of the major challenges in Fog computing paradigm. It plays a fundamental role to improve the performance of the whole system, by optimizing one or many objectives to map the tasks to the appropriate resources [19] in order to respond to the QoS required by the applications. At the same time, it aims to improve the degree of customer satisfaction [19].

Fog computing has a hierarchical architecture, where resources are distributed in the different layers of the paradigm. Each layer has its own characteristics and capacities. The scheduling problem in Fog computing presents a new aspect of scheduling compared to traditional scheduling in related paradigms.

In this paper, we present the most recent research work concerning the different scheduling methods and algorithms. The main purpose of these methods is to improve scheduling and optimize the performance of applications in Fog computing.

1.3 Related surveys and our contributions

In this section we present some surveys related to our work. The surveys of Naha et al. [8], Mouradian et al. [14] and Elavarasi et al. [20] have presented an overview of Fog computing: Architectures, Definitions, Research Directions and Challenges, state of the art and Fog applications. Mouradian et al. [14] presented the task scheduling algorithms and analyze them according to different criteria. In the paper [8], the authors discuss the existing research works and gaps in resource allocation and scheduling. In the survey [20], the authors compare various scheduling algorithms based on performance metrics. In another survey [21], the authors analyzed the research studies about task scheduling approaches in fog computing from 2015 to 2018. They proposed a classification of task scheduling approaches in two categories: static and dynamic.

In this survey, we summarize our main contributions as follows:

(1) Present a literature review of the various methods and scheduling algorithms recently proposed and used in Fog computing.

(2) Propose a taxonomy, which classifies the different methods and scheduling algorithms.

(3) Compare the different types of scheduling algorithms according to QoS criteria and their appropriate applications.

(4) Compare a number of tools used to evaluate the different proposed approaches.

1.4 Paper organization

The reminder of this article is organized as follows: in Section 2, we present a taxonomy of the different methods and scheduling algorithms used in Fog computing. Section 3 reviews the literature and analyses various existing research works in this field. Sections 4 compares the different algorithms and methods according to the QoS criteria, and it classifies them according to the appropriate type of application. In section 5, we present and compare the different scheduling algorithms and simulation tools used in research work. Section 6 discusses the techniques and methods used to improve scheduling. We conclude our paper in Section 7.

2. TAXONOMY OF SCHEDULING ALGORITHMS AND METHODS IN FOG COMPUTING

In this section, we propose a taxonomy of the different methods and scheduling algorithms in the Fog Computing environment as shown in Figure 2. This taxonomy is based on the classification of the various existing research works in the literature. It deals with a common objective the scheduling problem.

Our classification is divided into three major categories. This division is based on the nature of the method or algorithm used to improve scheduling. In the first category, we have grouped together the scheduling methods, based on optimization algorithms of an approximate nature, such as

heuristic algorithms like: "Min-Min and Greedy", and metaheuristic algorithms like "PSO, ACO, GA", and also the hybrid algorithms which are composed of several methods such as: "ACO-PSO, PSO-Min-Min, BLA-Greedy". The second category of scheduling methods, is concerned with scheduling optimizations based on the QoS constraints of an application, such as reduction of execution time, cost of resource usage and amount of energy consumed by the nodes of an application. Most of these proposed approaches use approximation algorithms based on Heuristic, Meta-heuristic and Hybrid, in order to improve the quality of the results. For the third category, we classified the scheduling methods that use data mining algorithms, in order to improve the performance of the scheduling.

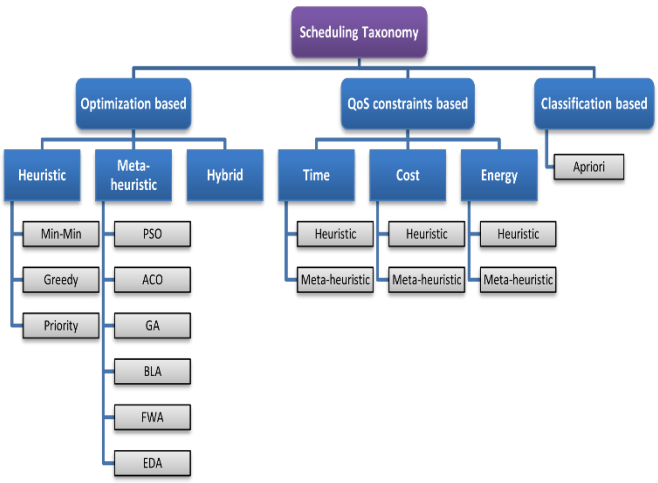


Figure 2. Proposed taxonomy of scheduling algorithms and methods in Fog Computing

In Table 2, we define the different acronyms used in this paper.

Table 2. List of important acronyms

Acronym	Definition
ACO	Ant Colony Optimization Algorithm
BLA	Bees Life Algorithm
EDA	Estimation of Distribution Algorithm
FWA	Fireworks Algorithm
GA	Genetic Algorithm
PSO	Particle Swarm Optimization
EDF	Earliest Deadline First
EFT	Earliest Finish Time
QoS	Quality of service
DAG	Directed Acyclic Graph

3. LITERATURE REVIEW

This section presents a literature review of the various recent research works proposed to demonstrate the different techniques and scheduling algorithms used in Fog computing to improve the performance of applications. According to the taxonomy proposed in this survey, we classify the different scheduling methods.

3.1 Scheduling based on optimization methods

These methods use heuristic, meta-heuristic or hybrid techniques. These proposed algorithms are compared to traditional algorithms in terms of performance in order to adapt them to Fog computing paradigm.

3.1.1 Heuristics algorithms based scheduling

Choudhari et al. [22] proposed a priority levels-based task scheduling algorithm (PLTS) in the Fog layer by combining two algorithms. The first one is an efficient resource allocation (ERA) algorithm proposed by Agarwal et al. [23]. It is implemented in the Fog layer and uses a three-layer architecture model (Client-Fog-Cloud). The second algorithm is a priority-based scheduling algorithm, proposed by Dakshayini et al. [24] and implemented in cloud computing environment.

PLTS algorithm starts by checking the availability of resources in the Fog nodes that satisfy the clients' requirements. If the resources are sufficient in this layer and the requests can be served by the deadline, the algorithm assigns a priority level to the client request and processes it later. However, if resources are not sufficient then it sends the requests to the Cloud layer.

The authors performed several simulation scenarios with CloudAnalyst tool. For each scenario, the results of the performance metrics of the algorithm proposed are compared to those of the following three policies: Optimize Response Time (ORT), Reconfigure Dynamically (RD), and Efficient Resource Allocation (ERA) [23]. This study showed that the proposed prioritized scheduling algorithm has reduced the response time and has considerably decreased the cost. However, it used the static priority settings in response to request traffic load.

3.1.2 Meta-heuristics and hybrid algorithms based scheduling

Wang et al. [25] proposed a task scheduling algorithm in Fog computing based on the improvement of the bio-inspired Firework Algorithm (FA) [26].

The authors proposed two contributions. First, they improved Firework Algorithm (I-FA) by the introduction of an explosion radius detection mechanism. Second, they used the proposed task scheduling algorithm (I-FASC) which takes into account the characteristics of the tasks and the resources. They have classified the tasks into three clusters according to expected storage space, expected time and expected bandwidth. Then, they integrated resources which are of three types: computing, storage and bandwidth.

The experimental results showed that the algorithm (I-FA) has an average explosion radius smaller than (FA) algorithm, and performed better than the other algorithms such as ACO-based algorithm (Rank-ACO) [27], double-fitness Genetic algorithm (DFGA) [28] and FA [26]. It reduces the number of iterations with the fastest convergence speed. The algorithm (I-FASC) reduces the task completion time compared to the other three algorithms: FSFC, Rank-ACO and DFGA. It allocates the resource in a more balanced way, which improves the performance of the entire system.

Xu et al. [29] proposed a method of scheduling tasks in the Fog-Cloud environment (LBP-ACS). This latter is based on the combination of two algorithms: Laxity-Based Priority (LBPA) [29] and Ant Colony System (ACS) [30]. The aim is to schedule the tasks' execution in a way to respect the deadlines, and to minimize the total energy consumption. The authors started with the algorithm (LBPA), which calculates the laxity time for each task and assigns a priority to the laxity (the smaller the laxity of the task, the higher the priority of the task). They also proposed a constrained optimization algorithm based on the Ant Colony System algorithm, which

chooses the right Cloud or Fog resources for the task based on its propriety.

After testing and evaluating the proposed method, they compared its performance to three other algorithms: Greedy for Energy (GFE), Heterogeneous Earliest Finish Time (HEFT) [31], and Differential Evolution Ant Colony Optimization (DEACO) [32]. The results showed that the proposed algorithm (LBP-ACS) outperformed the other algorithms. It balances the benefits between schedule length and energy consumption for Cloud or Fog resources, with Lower failure ratio rate.

The authors of the work [33] focused on task scheduling problem for Bag-of-Tasks applications in Cloud Fog computing environment. They proposed an algorithm called a Time-Cost aware Scheduling algorithm (TCaS) based on an evolutionary algorithm (Genetic Algorithm). They used the same Fitness Function to compare its performance (trade-off between the Makespan and Total Cost) to the following algorithms: Modified Particle Swarm Optimization (MPSO) [34], Bee Life Algorithm (BLA) [35] and a traditional simple Round Robin (RR) algorithm. They carried out several experiments in two different scenarios: one in a local Fog environment and another in a Cloud-Fog environment. The results show that for the first scenario TCaS has a better makespan and total cost; for the second scenario TCaS has a better makespan than the other algorithms, however MPSO has a better total cost than the other algorithms. They also demonstrated that (TCaS) is less convergent compared to the other algorithms (MPSO, BLA), however, it generates a more optimal solution.

Xu et al. [36] focused on scheduling Workflow in the Cloud-Fog environment. They presented a scheduling method that solves the mapping process between tasks and resources to minimize the makespan of Workflow and the cost. This method is an optimization of PSO algorithm. They introduce a new non-linear function (inertia weight), which enhances the global and local search capabilities of particles.

Bitam et al. [37] proposed a new bio-inspired algorithm for scheduling tasks, based on Bee Swarm optimization algorithm called Bees Life Algorithm (BLA) [35]. This new algorithm focuses on scheduling improvement in the fog computing environment. They designed a better allocation of tasks among the available Fog resources, by finding an optimal trade-off between CPU execution time and allocated memory. To achieve this goal, the authors proposed an optimization of the BLA algorithm using a Greedy approach to optimize the local search and improve the global solution. This way they reduced the latency and the cost to satisfy mobile users' requests.

All the authors' works discussed in this section are summarized and compared in Table 3.

Table 3. Comparison of meta-heuristics and hybrid algorithms based scheduling
(S: Simulation, E: Evaluation on a real server or platform)

Work	Evaluation	Tool	Infrastructure	Advantage	Limitations
[25]	E	Cloud Server	Fog/ Cloud	- Reduced average explosion radius - Fast convergence - Reduced task completion time - Load optimization	- Fog nodes' energy consumption is not considered
[29]	S	CloudSim		- Balance between schedule length and energy consumption for cloud or fog resources - Low failure ratio	- Not optimized for scheduling independent tasks
[33]	S	iFogSim	Fog/ Cloud	- Efficient trade-off between makespan and cost - TCaS is flexible in satisfying users' requirement with respect to highperformance processing and cost efficiency	- TCaS is more costly than MPSO in a Cloud–Fog environment.
[36]	S	Matlab	Fog/ Cloud	- Reduced Makespan and cost	-Lack of cost optimization
[37]	S	C++	Fog	- Reduced CPU execution time and memory allocation	- No dynamic job scheduling -No optimization of network bandwidth and cost

All works studied previously aim to improve the quality of scheduling and increase the performance of solution in terms of reducing latency, energy consumption, makespan and the cost. They are based on heuristics [22], meta-heuristics [33, 36] or hybrids [25, 29, 37] approaches.

It is noticed that the hybridization of the scheduling algorithms brought potential benefits where authors in [29] combined Greedy algorithm with ACO in order to have better performance and in [25] improved the firework algorithm by a genetic algorithm. In [37], the authors proposed a Bees Life Algorithm combined with Greedy approach to improve the local search process in order to reach the optimal individual. Other works favored the use of meta-heuristic algorithms such as in [33] where the authors propose a modification of GA in the Parental Selection part, this modification improved the value of the fitness function compared to the BLA and MPSO algorithms. In [36], the authors proposed an improvement of the PSO algorithm, by a novel update method of inertia weight that influences search capability of particles, and which facilitates to enhance the global search capability of particles.

3.2 QoS constraints-based scheduling

In this class of scheduling methods, the approaches proposed by the researchers focus on optimizing QoS constraints, in order to meet the performance requirements of the types of applications deployed on Fog computing.

3.2.1 Energy-efficiency based scheduling

Among the problems encountered in the Fog computing paradigm is the energy consumption management [9, 16] of the distributed nodes that must be efficient. The proposed works use the scheduling methods with the common objective of decreasing the energy dissipated by IoT nodes. In this

section, we present research works that deal with this type of problem.

Table 4 summarizes the Energy-Efficiency based Scheduling works.

Rahbari et al. [38] proposed a scheduling strategy called greedy knapsack-based scheduling (GKS) by combining the knapsack and Greedy algorithms. The knapsack algorithm was used for resource scheduling for allocation optimization of processing elements. Each application is a combination of a set of modules, and each module is assigned to a specific processing element. While a greedy algorithm was used to reduce the time delay in optimal allocation of resources to modules in the fog network. These two algorithms were used in order to optimize the objective of the knapsack problem by maximizing the profit and minimizing the weight. The results of different case studies of the proposed method GKS has reduced the number of application modules lunched by the processing elements of hosts in micro–Data centers, which has reduced energy consumption and cost at the same time.

Wan et al. [39] proposed an architecture for smart factory applications based on Fog Computing, which aims to solve the problem of energy consumption related to equipment workload. The authors proposed an energy-aware load balancing and scheduling method called (ELBS). It makes it possible to establish an energy model of the equipment deployed on the Fog nodes for load balancing. It has an objective function to be minimized for optimizing scheduling. The authors implemented an improved PSO algorithm in order to find better results. The multi-agent system is introduced, to realize dynamic scheduling of intelligent equipment using Fog nodes according to the load balancing scheduling strategy. The authors have performed experiments on a real environment, which presents a prototype platform of type intelligent manufacturing system. The results show that the proposed method has a better-balanced workload, and optimized the energy consumption.

Wang et al. [17] studied the task scheduling strategy in the Fog computing scenario. They proposed a task scheduling strategy based on a hybrid heuristic algorithm (HH). It deals with resource limitations and high energy dissipated by smart manufacturing devices. HH algorithm combines the advantages of two algorithms: the improved particle swarm optimization (IPSO) and the improved ant colony optimization (IACO). IPSO has a fast convergence and IACO has high precision characteristics to obtain the optimal solution of task scheduling. Based on experiment results, the proposed Hybrid Heuristic algorithm (HH) have outperformed other simple algorithms IPSO, IACO, Round-Robin. Also, the energy consumption is proportional to the completion time.

Luo et al. [40] proposed a new hierarchical architecture called multi-cloud to the existed multi-fog architecture based on containers technology. The multi-cloud aims to improve the resource utilization by Fog nodes and reduce the service delay. They also proposed a scheduling algorithm to deploy on the Fog nodes based on the energy balancing strategy. Its role is to control the transmission power of terminal devices, according to their energy levels, in order to extend the wireless sensor networks lifetime.

Furthermore, it aims to reduce the delay constraint of tasks through the collaboration of the Fog nodes and the Cloud according to the available resource threshold of the Fog node.

Wu et al. [41] proposed several models. They modeled the IoT system as a three-level model. The IoT application is modeled by a directed acyclic graph. They proposed a model to study the energy consumption of the overall three-tier IoT system. The DAG is divided into two parts, one part is assigned on the things tiers and processed locally, and the second part is sent to the Fog and Cloud tiers for further processing. The tasks assigned to the things tier are sorted by a heuristic method called bottom level (b-level) [42], and the tasks assigned to the Fog and Cloud tiers are sorted by the proposed Estimation of Distribution Algorithm with the partition operator (EDA-p). When these two processing sequences are determined, the tasks will be assigned to the nodes by the rule of Earliest Finish Time First (EFTF). The proposed scheduling algorithm EDA-p aims to achieve the trade-off between energy saving and shorten makespan. Its performance was evaluated using several cases of comparative studies. The experiment results show that the algorithm is effective, in reducing makespan and the energy consumption of devices, as well as extending the life time of IoT devices.

Table 4 recapitulates the previous discussed Energy-Efficiency based Scheduling works.

Several approaches and strategies have been used with scheduling algorithms for different applications to use energy in an efficient way. Wan et al. [39] proposed several techniques for Smart Factory type application to reduce the energy consumption by factory terminals equipment. Their scheduling method is based on load balancing mechanism. The scheduling method proposed by Wang and Li [17] is based on a reasonable allocation of resources.

Table 4. Comparison of energy efficiency based scheduling

Work	Evaluation	Tool	Infrastructure	Advantage	Limitations
[38]	S	iFogsim	Fog	- Reduction of energy consumption, cost and delay	- No performance comparison to other heuristic algorithms
[39]	E	Prototype Platform	Fog	- Balanced workload - Energy consumption optimized	- No makespan optimization
[17]	S	Matlab	Fog	- Reduced completion time and energy consumption - Improved reliability	- Requires more completion time and energy consumption than ISPO algorithm
[40]	E	Use a real machine with virtualization technology	Fog/ Cloud	-Terminal devices' energy is well balanced	-No comparison to other algorithms
[41]	S	C++	Fog/ Cloud	- Reduced energy consumption and makespan	- Scheduling applications with flexible or strict deadlines were not considered

In battery-based applications, where system life is a major issue, Luo et al. [40] have proposed a multi-cloud to multi-fog based on containers architecture, which uses a scheduling algorithm based on the energy balancing strategy, between terminal devices and an energy harvesting equipment. Another solution proposed by Wu et al. [41], uses scheduling algorithm to minimize the energy consumed by IoT devices.

All these strategies proposed previously, are used with scheduling algorithms, in order to improve the energy efficiency. Most of these proposed algorithms are of heuristic and meta-heuristic type [38, 40, 41] and hybrid [17, 39].

3.2.2 Time-efficiency based scheduling

One of the major problems encountered by these types of critical IoT applications, is the time constraint (or the real-time interaction) requiring application tasks to complete within an expected deadline. Next, we discuss the researches about different scheduling methods, which aim to reduce the applications' response time. In what follows, we will review some Time-Efficiency based Scheduling works and compare them in the Table 5.

Stavrinides et al. [43] proposed a new approach, which attempts to schedule computationally demanding tasks, with low communication requirements on the Cloud, and communication intensive tasks, with low computational demands on the Fog computing layer. By contrast other approaches perform the IoT tasks on the Fog computation layer. This approach is based on a hybrid Fog and Cloud-aware heuristic, for the dynamic scheduling of multiple real-time IoT workflows in a three-tiered architecture. This heuristic approach allows the scheduling of a ready task of a workflow on either Fog or Cloud layer based on its potential communication and computational requirement. The proposed scheduling strategy is divided in two phases. A task selection phase that prioritizes tasks according to the Earliest Deadline First (EDF) policy, and a VM selection phase, which allocates the selected task to VM using estimated Earliest Finish Time (EFT) policy.

The authors compared the performance of this approach with a proposed alternative version of Cloud-unaware scheduling strategy called Fog-EDF. The obtained results show that the Hybrid-EDF method reduces the deadline miss ratio in a significant way by comparison to Fog-EDF.

Auluck et al. [44] proposed an improvement in real-time task scheduling using the Fog computing, which is based on three-tiers architecture: local embedded, Fog and Cloud. This approach allows application tasks to be assigned to the appropriate tier for successful execution, which ensures minimal overall communication time. First of all, the authors divided the tasks into three categories according to their delay tolerance hard real-time, firm real-time and soft real-time. To improve the scheduling of these tasks, they proposed two types of algorithms.

The first algorithm is Static LFC (Local, Fog, Cloud), which statically assigns the tasks to be executed to the appropriate queue of different layers by the application of schedulability test, and the use of an optimization model. This leads to minimizing the total communication delay between Local, Fog and Cloud. The second algorithm is Self-Contained LFC (Local, Fog, Cloud), which schedules tasks without the use of schedulability test. It starts by scheduling the hard real-time type tasks on the local embedded processors. Then if the other types of tasks fail to meet their deadlines in this layer, they are scheduled in their appropriate layers with the use of an optimization model to minimize the total communication delay, which includes the tasks' constraint deadline. The authors implemented their approach based on EDF scheduling algorithm, and compared it to other algorithms based on the same scheduling algorithm, which uses one of the following: Embedded-Cloud, Embedded-Fog or Fog-Cloud. The obtained results showed that the proposed approach improves the throughput, success ratio and response time.

Mukherjee et al. [45] aim to minimize the failure probability, to meet the different delay deadlines for the tasks that have arrived at the fog. They considered the system to be composed of a set of Fog nodes and end-users that are uniformly and randomly distributed over the entire network. The end-users offload their entire tasks to a nearby Fog node referred to as primary Fog node. If the computing resources are not sufficient at the level of primary Fog node, it will offload these tasks to the other neighboring Fog nodes. They assign a priority to the tasks according to the value of delay-deadlines. They consider that each Fog node maintains two virtual queues namely high-priority queue and low-priority queue. The scheduling of these queues is done using Lyapunov optimization function. The authors run a series of experiments using Monte Carlo simulations. The obtained results show that the scheduling method which uses the offloading tasks and Lyapunov drift-plus-penalty function, improved reliability (how many tasks meet their deadlines) compared to random scheduling.

Aburukba et al. [46] modeled the problem of scheduling IoT devices' requests in edge layer and assign them the adequate resources available at both Fog and Cloud layers in order to reduce the latency in the hybrid architecture Fog-Cloud computing. They proposed a customized implementation of Genetic algorithms.

Table 5. Comparison of time-efficiency based scheduling

Work	Evaluation	Tool	Infrastructure	Advantage	Limitations
[43]	S	C++	Fog/ Cloud	- Optimized real-time communication	- Cloud layer's monetary cost is not optimized
[44]	S	iFogsim	Fog/ Cloud	- Improved the communication delay	- No improvement of Cloud layer's monetary cost
[45]	S	Simulator	Fog	- Improved reliability	- Scheduling policy is not optimized under different resource configuration
[46]	S	Fog		- Efficient solutions - Minimize latency - Maximize resource utilization	- No comparison to other techniques

Table 6. Comparison of cost efficiency-based scheduling

Work	Evaluation	Tool	Infrastructure	Advantage	Limitations
[47]	S	Java	Fog/ Cloud	- Found the best scheduling plan with lowest cost under given deadlines	- lack of cost optimization
[48]	S	C++	Fog/ Cloud	- Effectively use the complementary cloud resources, and take into account the deadline constraints and the cost in the scheduling	- No dynamic scaling techniques.
[49]	S	CloudSim	Fog/ Cloud	- Achieve better trade-off between the makespan and the cost. - Meet user QoS requirements.	- No power consumption optimization.

They included a procedure for penalizing infeasible solutions, which do not satisfy the constraints of the problem (each request is allocated to one and only one resource, and it must satisfy the deadline requirements), by reducing the probability of unsatisfactory chromosome selection.

The authors carried out experiments to determine the parameters such as the population size and maximum number of iterations, which have a direct impact on the quality of the solution. They also compared the proposed GA to an exact algorithm called Branch-and-Bound algorithm (B&B). These two algorithms gave almost the same result for the latency value; however, the B&B algorithm did not converge to a solution for problems with large size. This algorithm took almost two days to compute a solution for a problem with 10 requests and 3 resources. On the other hand, the proposed GA gives a solution within a period of less than one minute for the same problem. Once the suitable parameters were determined, a series of simulations for different scenarios have been performed, and compared with other traditional algorithms such as Waited Fair Queuing, Priority Strict Queuing, and Round Robin. The results demonstrate that the proposed algorithm gives better performance in terms of overall latency, and meeting the requests deadlines by comparison to other unoptimized algorithms. The experiments also show that Fog computing provides better service latency than using only Cloud computing.

The approaches discussed in this section, have common objectives, which are to minimize latency, meet deadlines, and satisfy the requests of real-time applications. Several techniques and strategies offloaded tasks towards the most efficient nodes [44, 45], respected the deadline and minimized the delay. In the study [45], the authors detailed their scheduling strategy, and explained how to choose the appropriate node to offload the task.

The second technique, used in ref. [43-45], leverages priority-based algorithms, by favoring tasks that have the earliest deadline. For the scheduling algorithms, we observe that the authors [43, 44] use the priority-based algorithm Earliest Deadline First (EDF), which is the most used algorithm in the scheduling of real-time systems. The authors of the work in [45] proposed a new scheduling method based on Lyapunov optimization function. A new scheduling method [46] to minimize latency is based on a genetic algorithm. This approach reduces the latency by comparison to non-optimized algorithms.

3.2.3 Cost Efficiency based scheduling

The applications, deployed on Fog computing nodes, use several types of resources (Bandwidth, CPU, Storage, etc.) from different layers of the paradigm, in order to perform their tasks. The massive use of these resources depending on the nature of the applications generates additional costs. One of the advantages of this paradigm is to reduce the use of paid resources. For this purpose, we need efficient scheduling algorithms, which share the workload on the different available nodes in order to reduce the cost.

In this section, we review and compare (see Table 6) few approaches designed for cost efficient management.

Ding et al. [47] proposed a cost-effective scheduling strategy for multi-workflow with time constraints in Fog computing. This scheduling strategy uses multi-layer resources of Fog and Cloud computing called (CTSF). The proposed algorithm is based on Particle Swarm Optimization (PSO), which allows tasks to be allocated to adequate resources, and uses the fitness function with an objective of finding a minimum value of resource execution cost under given deadlines. In the case where multiple tasks are allocated to the same resource at the same time, a second algorithm called Min-Min algorithm is used to resolve the resource allocation conflict.

In order to evaluate the performance of the proposed scheduling strategy, two evaluation aspects have been carried out. First, the CTSF algorithm is compared to two other strategies, which use the same PSO and Min-Min algorithms, however they use the resources from a single layer either Fog or Cloud. Second, they evaluated the CTSF algorithm using different resource conflict resolution algorithms in multi-layer Fog and Cloud resources. Based on the obtained results, it is recommended to execute tasks that require large workload and small data on cloud servers because of the communication time saving. However, tasks with small workload and large data set should be executed on Fog nodes. The CTSF strategy can find the best scheduling plan with lowest cost under given deadlines, using the PSO and Min-Min algorithms, by reducing the execution time of conflicting tasks.

Stavrinides et al. [48] proposed a real-time scheduling strategy for the tasks of the multiple workflows, coming from IoT to Fog computing and which uses Cloud resources as a complement. This strategy is based on the trade-off between performance and monetary cost. It consists of two stages. A task selection stage, allows to prioritize the tasks of the global waiting queue for the central scheduler in Fog layer, according to their deadline using the Earliest Deadline First (EDF) policy. In the case where several tasks have the same priority, the task with highest average computational cost has highest priority. A virtual machine selection stage, which allocates the tasks, selected by the scheduler, to the adequate virtual machine of the Fog or Cloud tier, according to the minimum value given by the proposed objective function. This function takes into account the sum of two parameters (the estimated finish time and the estimated monetary cost of resource usage). Each one of these parameters is assigned a weight indicating its contribution factor. This weight is calculated using a proposed scheduling heuristic.

The authors compared the performance of their proposed approach to a baseline policy MinEFT-Fog that uses only the

Fog layer's resources. The obtained results show that the proposed approach assigns tasks that require high computation but low communication to virtual machine in the Cloud. However, it assigns tasks with low computation and intensive communication to Fog virtual machines. This reduces communication costs and saves the average monetary cost compared to the baseline policy.

Pham et al. [49] proposed a cost and makespan aware scheduling algorithm called (CMaS). The objective of this algorithm is to achieve a good trade-off between the application execution time and the cost, for the use of Cloud resources and satisfy the user defined deadline constraints. The algorithm is divided into three phases. The first phase is to order the tasks based on the length of the critical path. The second phase is the node selection that allows the assignment of each task to an appropriate processing node on the Cloud or Fog, to achieve the optimal value of a utility function. The third phase is deadline-based task reassignment, which limits the deadline violation by reallocating critical tasks to the best processing nodes. This can reduce the completion time of each critical task.

In order to demonstrate the performance of the proposed algorithm, the authors performed two types of evaluations. The first one is the evaluation of the CMaS algorithm efficiency compared with other algorithms. The results show that the algorithm can achieve better trade-off between the makespan and the cost of task execution than other methods. The second evaluation concerns the deadline-based task reassignment. The results show that the deadline-based CMaS algorithm also gave a better performance in terms of makespan. It guarantees the end of application execution before the predefined deadline to satisfy the user QoS requirements.

After detailing some of the works in Cost Efficiency-based Scheduling category, we observed that authors in [47, 48] have proposed scheduling methods and strategy for multi-workflow standard applications. While the author in [49] applied these methods on workflow standard applications. Both methods are interested in minimizing two constraints at the same time. The aim is to reduce the cost of using resources. The execution time is taken in consideration by reducing the makespan [47, 49] or respecting the deadlines for real-time applications [48]. For the cost calculation, both works [48, 49] considered the type of resources and to which layers of the paradigm belong (Fog or Cloud), however, the work in [47] and the work of previous sections such as [22, 36, 37] did not optimize the cost based on the type of resources.

The work in Ding et al. [47] is based on a hybrid type optimization algorithm (PSO and Min-Min) to improve the quality of the result. Stavrinides et al. [48], Pham et al. [49] incorporated priority algorithms in their approaches to respond efficiently to the deadline constraint.

3.3 Classification-based scheduling

Liu et al. [50] proposed a new data mining classifications-based scheduling approach. They proposed a Task Scheduling algorithm in Fog Computing (TSFC) based on a new improvement of the traditional Apriori algorithm called I-Apriori. The main TSFC algorithm process is divided into two steps. First, it uses the I-Apriori algorithm to generate association rules of nodes and tasks sets from scheduling transaction set. Then, these association rules are used with the TSFC algorithm to get the task scheduling relationship between the Fog nodes and the tasks. After comparing the TSFC algorithm to other heuristic algorithms, the results show that, this new scheduling approach is more efficient than the Minimum Completion Time algorithm (MCT) [51], Minimum Execution Time algorithm (MET) [51] and MIN-MIN algorithm [52], in terms of completion time and waiting time. However, TSFC algorithm did not ensure the multi-layered task scheduling and the scheduling optimization for other QoS parameters.

4. COMPARISON OF SCHEDULING ALGORITHMS AND METHODS ACCORDING TO QOS CRITERIA AND THEIR TYPES OF APPLICATIONS

In this section, we present comparisons of many algorithms for Scheduling problems that have been mentioned earlier in terms of QoS criteria (see Table 7) and the type of appropriate applications (see Table 8).

4.1 Comparison based on QoS criteria

The QoS criteria used in this study are summarized as follows:

(1) **Cost:** the monetary cost of using resources (Processors, Memory, Storage, Bandwidth, etc.).
(2) **Energy:** the energy consumption for nodes' resources.
(3) **Workload Ratio:** the ratio of the amount tasks to be performed by a node.
(4) **Resources Usage:** the resource (Processors, Memory, Storage, Bandwidth, etc.) utilization rate of a node.
(5) **Throughput:** the number of tasks that complete their execution in a time interval [45].
(6) **Reliability:** the success rate of a task execution under the constraints of the maximum tolerance time.
(7) **Response Time:** the total time it takes between a service request and responding to that request.
(8) **Missed-Deadline:** the ratio of the number of tasks that did not complete their execution within their deadlines.
(9) **Makespan:** the time of processing all tasks.
(10) **Delay:** the waiting time or the time required for a task to travel from the source to the destination on which it is executed [45].

4.2 Classification according to the type of applications

In the literature, several studies have proposed a variety of IoT applications' classifications according to their characteristics and QoS requirements. We propose in this section a new classification of the different scheduling methods mentioned previously based on the adequate type of IoT applications.

Table 7. A comparison of scheduling algorithms and methods based on QoS criteria

Category	References	Cost	Energy	Workload Ratio	Resources Usage	Throughput	Reliability	Response Time	Missed - Deadline	Makespan	Delay
Heuristic based	Choudhari et al. [22]	✓						✓			
Meta-heuristic based	Wang et al. [25]				✓					✓	
	Xu et al. [29]		✓						✓	✓	
	Nguyen et al. [33]	✓								✓	
	Xu et al. [36]	✓								✓	
	Bitam et al. [37]				✓					✓	
Energy Efficiency based	Rahbari et al. [38]	✓	✓		✓						
	Wan et al. [39]		✓	✓							
	Wang et al. [17]		✓				✓			✓	
	Luo et al. [40]		✓								✓
	Wu et al. [41]		✓							✓	
Time Efficiency based	Stavrinides et al. [43]	✓							✓		
	Auluck et al. [44]					✓		✓	✓		✓
	Mukherjee et al. [45]						✓		✓		
	Aburukba et al. [46]								✓		✓
Cost Efficiency based	Ding et al. [47]	✓								✓	
	Stavrinides et al. [48]	✓							✓		
	Pham et al. [49]	✓								✓	
Classification based	Liu et al. [50]									✓	✓

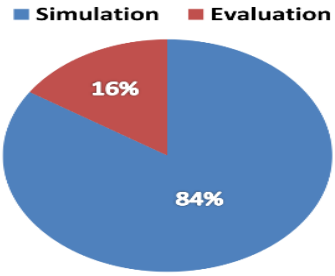
Table 8. A classification according to the type applications

References	Smart home	Wearables	Smart Farming	Smart City	Autonomous Vehicles	Healthcare	Smart Industry	Smart traffic	Gaming, AR and VR
Choudhari et al. [22]	✓								✓
Wang et al. [25]	✓	✓	✓		✓				
Xu et al. [29]	✓	✓	✓		✓	✓	✓	✓	✓
Nguyen et al. [33]	✓			✓				✓	
Xu et al. [36]	✓			✓				✓	
Bitam et al. [37]	✓	✓	✓		✓				
Rahbari et al. [38]	✓	✓	✓	✓	✓		✓	✓	✓
Wan et al. [39]	✓	✓	✓				✓		
Wang et al. [17]	✓	✓	✓		✓	✓	✓	✓	
Luo et al. [40]	✓	✓	✓		✓	✓	✓	✓	
Wu et al. [41]	✓	✓	✓				✓		
Stavrinides et al. [43]	✓			✓	✓	✓	✓	✓	✓
Auluck et al. [44]	✓				✓	✓	✓	✓	✓
M. Mukherjee et al. [45]	✓				✓	✓	✓	✓	✓
Aburukba et al. [46]	✓				✓	✓	✓	✓	✓
Ding et al. [47]	✓			✓				✓	
Stavrinides et al. [48]	✓			✓	✓	✓	✓	✓	✓
Pham et al. [49]	✓			✓				✓	
Liu et al. [50]	✓				✓	✓	✓	✓	✓

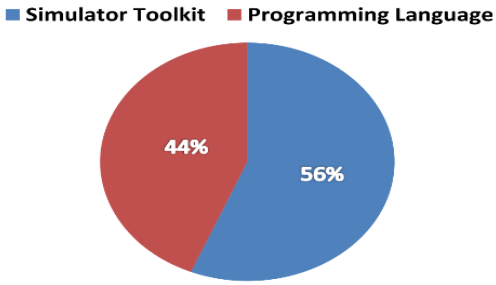
5. COMPARISON OF SIMULATION TOOLS AND TYPES OF SCHEDULING ALGORITHMS USED

In this section we present, a comparison of the use of different types of scheduling algorithms and also the simulation tools used by the researchers in their work.

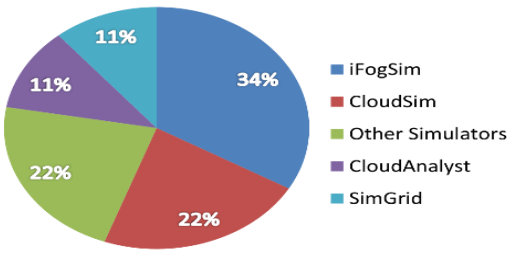
5.1 Simulation tools comparison



(a) Simulation VS Evaluation



(b) Programming Language VS Simulator Toolkit



(c) The list of Simulator Toolkit

Figure 3. Programming languages and simulation toolkit used in the experiment

Different Simulator Tool Kits Description:

(1) **CloudSim:** is an open-source extensible simulation toolkit, that enables modeling and simulation of Cloud computing systems and application provisioning environments. It allows the evaluation of the performance of resource provisioning and application scheduling techniques, under different usage and infrastructure scenarios [53].

(2) **iFogSim:** is a simulation toolkit developed based on the fundamental framework of CloudSim. It extends the abstraction of basic CloudSim classes, and offers scopes to simulate customized Fog computing environment with large number of Fog nodes and IoT devices. In addition, it facilitates evaluation of end-to-end latency, network congestion, power usage, operational expenses and QoS satisfaction [54].

(3) **CloudAnalyst:** extends the functionalities of CloudSim. This tool supports visual modeling and simulation of large-scale applications that are deployed on Cloud Infrastructures. The main objectives of CloudAnalyst are to separate the simulation experimentation exercise from a programming exercise. It allows description of application workloads, including users' geographical location information generating traffic and location of data centers, number of users and data centers, and number of resources in each data center. Using this information, CloudAnalyst generates information about response time of requests, processing time of requests, and other metrics [55].

(4) **SimGrid:** it is an open-source framework, which allows the simulation of distributed computer system used in studies on Grids, Clusters, High Performance Computing, P2P systems and Fog Computing [56].

(5) **Matlab:** is a tool often used in scientific fields to perform numerical calculations. It manipulates matrices, displays curves and data, and writes scripts and algorithms.

(6) **C++:** is a programming language, used in the development of applications that require high performance.

(7) **Java:** a general-purpose object-oriented programming language.

Based on our study of literature, we observe that the evaluations in a real environment present only 14% of experimental work. On the other hand, the authors tend to use simulation tools in their experimental work which presents 84%. This is due to the high cost and difficulty of implementation caused by the evaluation in a real environment (see Figure 3.a) [57].

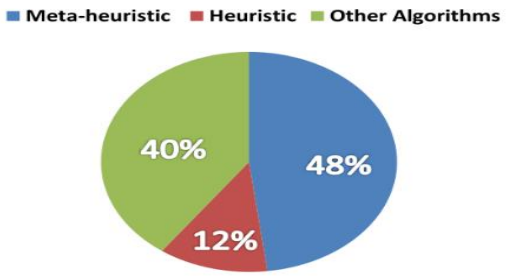
These simulations are carried out by two different methods, either by programming or by the use of a Simulator Toolkit or Framework. Most of the simulations (56%) are carried out using a Simulator Toolkit (see Figure 3.b), because these tools facilitate the development, modeling and evaluation of the model to be simulated by using the functionalities offered by the Toolkit. This allows the researchers to focus on the simulation and analysis of the proposed approach's behavior, analyze the results obtained in the various simulation scenarios [57, 58], and not focus on the details of programming the model to be simulated.

In the literature, the authors use several types of Simulator Toolkit as shown in. The majority of studies cited previously have generally used two types of Simulator Toolkit, the first is CloudSim specialized in the simulation of the scheduling in the Cloud Computing environment [59], and the second iFogSim is specialized in scheduling simulation in the Fog Computing environment as shown in Figure 3.c [54, 59].

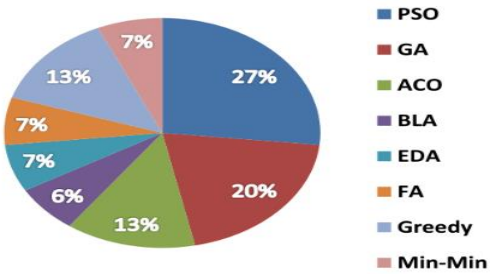
5.2 Comparison of the type of algorithms used

According to the work cited above, we notice that most algorithms used to solve the scheduling problem, which is of NP-hard nature [54], are Heuristic or Meta-heuristic type (88%) as shown in Figure 4.a. This is due to the fact that this kind of algorithms have the ability to provide solutions close to the optimum within a reasonable time [60-62]. By comparison, exact algorithms give optimal solutions but in a very inefficient way. In the studies of the authors cited in this paper, we observe that the meta-heuristic algorithms are used more than heuristic algorithms (see Figure 4.a), because they provide better results closer to the optimal solution [60]. Among these meta-heuristic algorithms, we see that the most used algorithms are: PSO (27%), GA (20%) and ACO (13%). (see Figure 4.b). 67% the scheduling approaches, which use meta-heuristic algorithms are hybrid as shown in Figure 4.c. The hybridization of these meta-heuristic algorithms,

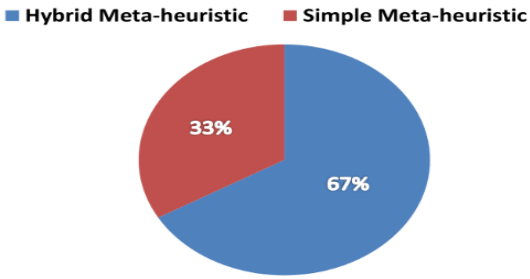
improves the performance and the quality of the solution generated by the algorithm, as well as the speed of convergence [60, 63].



(a) The use of Heuristic and Meta-heuristic Algorithms



(b) The list of Heuristic and Meta-heuristic Algorithms



(c) The use of type of Meta-heuristic Algorithms

Figure 4. The type of algorithms used in the scheduling

6. DISCUSSION

In general, the scheduling problem is NP-hard problem which requires optimization methods to solve it. For this, approximation algorithms [60] based on Heuristic and Meta-heuristic algorithms are used to get a semi optimal solution, which are very close to an exact solution in a reasonable time. For instance, in ref. [46], the authors performed a comparison in terms of solution quality and execution time between a Meta-heuristic algorithm (GA) and an exact algorithm (Branch and bound (B&B)). After reducing the size of the problem for the exact algorithm (B&B), it takes 2 days to calculate the result by comparison the GA finds a solution, which is very close the solution found by the exact algorithm, in less than one minute. According to the comparison shown in the previous section, we observed that most of the scheduling approaches use meta-heuristic and hybrid algorithms because they give near optimal solution in a very efficient way.

Liu et al. [50] proposed another type of scheduling approach, instead of using approximation algorithms, they used an algorithm based on the classification techniques of Data mining called the Apriori algorithm, which obtained acceptable results in a reduced time complexity. One of the objectives of these scheduling methods and algorithms is to be used in the optimization of the QoS of the applications, so that they meet the performance requirements of the applications such as the optimization of energy consumption, cost and latency. The works of the researchers cited previously combine scheduling algorithms based on optimization techniques to respond to this type of problem and effectively manage the QoS constraints required by these applications:

(1) Energy consumption reduction: the authors reduce the number of nodes used to run the applications. They use the load balancing method based on energy awareness. This efficient energy management increases the lifespan of the IoT applications battery-based, and reduces the cost of power consumption by nodes.

(2) For real-time and critical application, meeting the deadlines and reducing the response time are important. For this purpose, the authors use the techniques of prioritized queues and offloading the tasks in the most efficient nodes.

(3) To reduce costs, the authors minimize the use of expensive nodes such as cloud nodes to accomplish different tasks. They favor the use of the local node on the fog layer, which at the same time reduces the communication to the cloud without affecting the application performance.

Finally, the evaluation of the different scheduling approaches proposed by the researchers, most of them use Simulator ToolKit tools such as CloudSim or iFogSim, which facilitates simulation and analysis of results.

7. CONCLUSIONS

In this paper, we have presented a survey on different scheduling algorithms and methods used in Fog computing. Through this study, we have reviewed, analyzed and compared different algorithms and approaches proposed to improve scheduling in this paradigm. Also, we have classified these methods in several categories according to their nature. We have deducted that most of the algorithms and scheduling methods proposed are approximation algorithms of meta-heuristic-based type. These types of algorithms are used to optimize the system performance by decreasing energy consumed by IoT devices and minimizing the cost/latency of transmissions. Consequently, they can meet the workload and performance required by critical IoT applications deployed on Fog computing.

In future, based on this study we plan to propose a new approach in this thematic by combining edge and fog layers algorithms to efficiently optimize the whole system.

REFERENCES

[1] Perera, C., Zaslavsky, A., Christen, P., Georgakopoulos, D. (2014). Context aware computing for the internet of things: A survey. IEEE Commun. Surv. Tutorials, 16(1): 414-454. https://doi.org/10.1109/SURV.2013.042313.00197

[2] Hassija, V., Chamola, V., Saxena, V., Jain, D., Goyal, P., Sikdar, B. (2019). A survey on IoT security: Application areas, security threats, and solution architectures. IEEE Access, 7: 82721-82743. https://doi.org/10.1109/ACCESS.2019.2924045

[3] Evans, D. (2011). How the next evolution of the internet

is changing everything. cisco white paper. https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf, accessed on Nov. 04, 2020.
[4] Rao, T., Khan, M.A., Maschendra, M., Kumar, M.K. (2015). A paradigm shift from cloud to Fog Computing. In IJCSET, 385-389. http://www.ijcset.net
[5] Bonomi, F., Milito, R., Zhu, J., Addepalli, S. (2012). Fog computing and its role in the internet of things. In Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing - MCC'12, p. 13. https://doi.org/10.1145/2342509.2342513
[6] Fog computing and the internet of things: Extend the cloud to where the things are, cisco white paper. 2015. http://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf, accessed on Nov. 04, 2020.
[7] "OpenFog Reference Architecture for Fog Computing," 2017. [Online]. Available: https://www.iiconsortium.org/pdf/OpenFog_Reference_Architecture_2_09_17.pdf, accessed on Nov. 04, 2020.
[8] Naha, R.K., Garg, S., Georgakopoulos, D., Jayaraman, P.P., Gao, L.X., Xiang, Y., Ranjan, R. (2018). Fog computing: Survey of trends, architectures, requirements, and research directions. IEEE Access, 6: 47980-48009. https://doi.org/10.1109/ACCESS.2018.2866491
[9] Hu, P., Dhelim, S., Ning, H., Qiu, T. (2017). Survey on fog computing: architecture, key technologies, applications and open issues. J. Netw. Comput. Appl., 98: 27-42. https://doi.org/10.1016/j.jnca.2017.09.002
[10] Dastjerdi, A.V., Gupta, H., Calheiros, R.N., Ghosh, S.K., Buyya, R. (2016). Fog Computing: Principles, architectures, and applications. Internet Things Princ. Paradigms, 61-75. https://doi.org/10.1016/B978-0-12-805395-9.00004-6
[11] Nadeem, M.A., Saeed, M.A. (2016). Fog computing: An emerging paradigm. In 2016 Sixth International Conference on Innovative Computing Technology (INTECH), pp. 83-86. https://doi.org/10.1109/INTECH.2016.7845043
[12] Sarkar, S., Misra, S. (2016). Theoretical modelling of fog computing: a green computing paradigm to support IoT applications. IET Networks, 5(2): 23-29. https://doi.org/10.1049/iet-net.2015.0034
[13] Luan, T.H., Gao, L.X., Li, Z., Xiang, Y., Wei, G.Y., Sun, L. (2015). Fog computing: focusing on mobile users at the edge, 1-11. arXiv:1502.01815.
[14] Mouradian, C., Naboulsi, D., Yangui, S., Glitho, R., Morrow, M., Polakos, P. (2017). A comprehensive survey on Fog Computing: State-of-the-art and research challenges. IEEE Commun. Surv. & Tutorials, 20(1): 416-464. https://doi.org/10.1109/COMST.2017.2771153
[15] Dang, L.M., Piran, M.J., Han, D., Min, K., Moon, H. (2019). A survey on internet of things and cloud computing for healthcare. Electronics, 8(7): 768. https://doi.org/10.3390/electronics8070768
[16] Prakash, P, Darshaun, K.G., Yaazhlene, P, Ganesh, M.V., Vasudha, B. (2017). Fog Computing: Issues, challenges and future directions. Int. J. Electr. & Comput. Eng., 7(6). https://doi.org/10.11591/ijece.v7i6.pp3669-3673
[17] Wang, J., Li, D. (2019). Task scheduling based on a hybrid heuristic algorithm for smart production line with Fog Computing. Sensors, 19(5): 1023. https://doi.org/10.3390/s19051023
[18] Pinedo, M.L. (2012). Scheduling: Theory, Algorithms, and Systems. Springer. https://doi.org/10.1007/978-1-4614-2361-4
[19] Mon, M.M., Khine, M.A. (2019). Scheduling and load balancing in cloud-fog computing using swarm optimization techniques: A survey. In Seventeenth International Conference on Computer Applications (ICCA 2019), pp. 8-14.
[20] Elavarasi, R., Silas, S. (2019). Survey on job scheduling in Fog Computing. In 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), pp. 580-583. https://doi.org/10.1109/ICOEI.2019.8862651
[21] Hosseinioun, P., Kheirabadi, M., Reza, S., Tabbakh, K., Ghaemi, R. (2020). A Task scheduling approaches in fog computing: A survey. Trans. Emerg. Telecommun. Technol. https://doi.org/10.1002/ett.3792
[22] Choudhari, T., Moh, M., Moh, T.-S. (2018). Prioritized task scheduling in fog computing. In Proceedings of the ACMSE 2018 Conference, pp. 1-8. https://doi.org/10.1145/3190645.3190699
[23] Agarwal, S., Yadav, S., Yadav, A.K. (2016). An efficient architecture and algorithm for resource provisioning in fog computing. Int. J. Inf. Eng. Electron. Bus., 8(1): 48-61. https://doi.org/10.5815/ijieeb.2016.01.06
[24] Dakshayini, D.M., Guruprasad, D.H.S. (2011). An optimal model for priority-based service scheduling policy for cloud computing environment. Int. J. Comput. Appl., 32(9): 23-29.
[25] Wang, S., Zhao, T., Pang, S. (2020). Task scheduling algorithm based on improved firework algorithm in Fog Computing. IEEE Access, 8: 32385-32394. https://doi.org/10.1109/ACCESS.2020.2973758
[26] Tan, Y., Zhu, Y. (2010). Fireworks Algorithm for Optimization. In: Tan Y., Shi Y., Tan K.C. (eds) Advances in Swarm Intelligence. ICSI 2010. Lecture Notes in Computer Science, vol 6145. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-13495-1_44
[27] Edward, N., Elcock, J. (2018). An efficient task scheduling algorithm for heterogeneous multiprocessing environments. In 2018 International Conference on Information and Computer Technologies (ICICT), pp. 101-106. https://doi.org/10.1109/INFOCT.2018.8356849
[28] Li, J.F., Peng, J. (2011). Task scheduling algorithm based on improved genetic algorithm in cloud computing environment. Jisuanji Yingyong/ J. Comput. Appl., 31(1): 184-186. https://doi.org/10.3724/SP.J.1087.2011.00184
[29] Xu, J., Hao, Z., Zhang, R., Sun, X. (2019). A method based on the combination of laxity and ant colony system for cloud-Fog Task scheduling. IEEE Access, 7: 116218-116226. https://doi.org/10.1109/ACCESS.2019.2936116
[30] Pei, Y., Wang, W., Zhang, S. (2012). Basic ant colony optimization. In 2012 International Conference on Computer Science and Electronics Engineering, pp. 665-667. https://doi.org/10.1109/ICCSEE.2012.178
[31] Topcuouglu, H., Hariri, S., Wu, M. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans. Parallel Distrib. Syst., 13(3): 260-274. https://doi.org/10.1109/71.993206

[32] Xiangyin, Z., hang, H.D., Jiqiang, J. (2008). DEACO: Hybrid ant colony optimization with differential evolution. In 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), pp. 921-927. https://doi.org/10.1109/CEC.2008.4630906

[33] Nguyen, B.M., Thi, H., Thanh, B., The Anh, T., Bao Son, D. (2019). Evolutionary algorithms to optimize task scheduling problem for the IoT based bag-of-tasks application in cloud–fog computing environment. Appl. Sci., 9(9): 1730. https://doi.org/10.3390/app9091730

[34] Tian, D., Shi, Z. (2018). MPSO: Modified particle swarm optimization and its applications. Swarm Evol. Comput., 41: 49-68. https://doi.org/10.1016/j.swevo.2018.01.011

[35] Bitam, S., Khider, M. (2012). Bees Life algorithm for job scheduling in cloud computing. In International Conference on Computing and Information Technology (ICCIT), pp. 186-191.

[36] Xu, R., Wang, Y.G., Cheng, Y.L., Zhu, Y.W., Xie, Y., Sani, A.S., Yuan, D. (2019). Improved particle swarm optimization based workflow scheduling in cloud-fog environment. In Business Process Management Workshops, pp. 337-347. https://doi.org/10.1007/978-3-030-11641-5_27

[37] Bitam, S., Zeadally, S., Mellouk, A. (2018). Fog computing job scheduling optimization based on bees swarm. Enterp. Inf. Syst., 12(4): 373-397. https://doi.org/10.1080/17517575.2017.1304579

[38] Rahbari, D., Nickray, M. (2019). Low-latency and energy-efficient scheduling in fog-based IoT applications. Turkish J. Electr. Eng. Comput. Sci., 27: 1406-1427. https://doi.org/10.3906/elk-1810-47

[39] Wan, J., Chen, B., Wang, S., Xia, M., Li, D., Liu, C. (2018). Fog Computing for energy-aware load balancing and scheduling in smart factory. IEEE Trans. Ind. Informatics, 14(10): 4548-4556. https://doi.org/10.1109/TII.2018.2818932

[40] Luo, J., Yin, L.X, Hu, J.Y., Wang, C., Liu, X., Fan, X., Luo, H.B. (2018). Container-based fog computing architecture and energy-balancing scheduling algorithm for energy IoT. Futur. Gener. Comput. Syst., 97: 50-60. https://doi.org/10.1016/j.future.2018.12.063

[41] Wu, C., Li, W., Wang, L., Zomaya, A. (2018). Hybrid evolutionary scheduling for energy-efficient fog-enhanced internet of things. IEEE Trans. Cloud Comput. https://doi.org/10.1109/TCC.2018.2889482

[42] Kwok, Y.K., Ahmad, I. (1999). Static scheduling algorithms for allocating directed task graphs to multiprocessors. ACM Comput. Surv., 31(4): 406-471. https://doi.org/10.1145/344588.344618

[43] Stavrinides, G.L., Karatza, H.D. (2019). A hybrid approach to scheduling real-time IoT workflows in fog and cloud environments. Multimed. Tools Appl., 78(17): 24639-24655. https://doi.org/10.1007/s11042-018-7051-9

[44] Auluck, N., Azim, A., Fizza, K. (2019). Improving the schedulability of real-time tasks using Fog Computing. IEEE Trans. Serv. Comput., pp. 1-1. https://doi.org/10.1109/TSC.2019.2944360

[45] Mukherjee, M., Guo, M., Lloret, J., Iqbal, R., Zhang, Q. (2017). Deadline-aware fair scheduling for offloaded tasks in Fog Computing with inter-fog dependency. IEEE Commun. Lett., 24(2): 307-311. https://doi.org/10.1109/LCOMM.2019.2957741

[46] Aburukba, R.O., AliKarrar, M., Landolsi, T., El-Fakih, K. (2020). Scheduling internet of things requests to minimize latency in hybrid fog–cloud computing. Futur. Gener. Comput. Syst., 111: 539-551. https://doi.org/10.1016/j.future.2019.09.039

[47] Ding, R., Li, X., Liu, X., Xu, J. (2019). A Cost-Effective Time-Constrained Multi-workflow Scheduling Strategy in Fog Computing. In: Liu X. et al. (eds) Service-Oriented Computing – ICSOC 2018 Workshops. ICSOC 2018. Lecture Notes in Computer Science, vol 11434. Springer, Cham. https://doi.org/10.1007/978-3-030-17642-6_17

[48] Stavrinides, G.L., Karatza, H.D. (2019). Cost-effective utilization of complementary cloud resources for the scheduling of real-time workflow applications in a fog environment. In 2019 7th International Conference on Future Internet of Things and Cloud (FiCloud), pp. 1-8. https://doi.org/10.1109/FiCloud.2019.00009

[49] Pham, X.Q., Man, N.D., Tri, N.D.T., Thai, N.Q., Huh, E.N. (2017). A cost- and performance-effective approach for task scheduling based on collaboration between cloud and fog computing. Int. J. Distrib. Sens. Networks, 13(11): 155014771774207. https://doi.org/10.1177/1550147717742073

[50] Liu, L., Qi, D., Zhou, N., Wu, Y. (2018). A task scheduling algorithm based on classification mining in Fog Computing environment. Wirel. Commun. Mob. Comput. 1-11. https://doi.org/10.1155/2018/2102348

[51] Maheswaran, M., Ali, S., Siegel, H.J., Hensgen, D., Freund, R.F. (1999). Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. J. Parallel Distrib. Comput., 59(2): 107-131. https://doi.org/10.1006/jpdc.1999.1581

[52] Braun, T. D., Siegel, H.J., Beck, N., Bölöni, L.L., Maheswaran, M., Reuther, A.I., Robertson, J.P., Theys, M.D., Yao, B., Hensgen, D., Freund, R.F. (2001). A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. J. Parallel Distrib. Comput., 61(6): 810-837. https://doi.org/10.1006/jpdc.2000.1714

[53] Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C. A.F., Buyya, R. (2011). CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw. Pract. Exp., 41(1): 23-50. https://doi.org/10.1002/spe.995

[54] Mahmud, M., Buyya, R. (2019). Modeling and simulation of fog and edge computing environments using iFogSim Toolkit. Fog and Edge Computing: Principles and Paradigms Principles and Paradigms, 433-465. https://doi.org/10.1002/9781119525080.ch17

[55] Wickremasinghe, B., Calheiros, R.N., Buyya, R. (2010). CloudAnalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications. In 2010 24th IEEE International Conference on Advanced Information Networking and Applications, pp. 446-452. https://doi.org/10.1109/AINA.2010.32

[56] SimGrid. https://simgrid.org/doc/latest/, accessed on Nov. 04, 2020.

[57] Suryateja, P.S. (2016). A comparative analysis of cloud simulators. Int. J. Mod. Educ. Comput. Sci., 8(4): 64-71. https://doi.org/10.5815/ijmecs.2016.04.08

[58] Byrne, J., Svorobej, S., Giannoutakis, K.M., Tzovaras,

D., Byrne, P.J., Östberg, P.O., Gourinovitch, A., Lynn, T. (2010). A review of cloud computing simulation platforms and related environments. In Proceedings of the 7th International Conference on Cloud Computing and Services Science, pp. 679-691. https://doi.org/10.5220/0006373006790691

[59] Kumar, R., Sahoo, G. (2014). Cloud computing simulation using CloudSim. Int. J. Eng. Trends Technol., 8(2): 82-86. https://doi.org/10.14445/22315381/IJETT-V8P216

[60] Jarboui, B., Siarry, P., Teghem, J. (2013). Metaheuristics for Production Scheduling. John Wiley & Sons, Inc. https://doi.org/10.1002/9781118731598

[61] Rabadi, G. (2016). Heuristics, metaheuristics and approximate methods in planning and scheduling. In International Series in Operations Research & Management Science, 236: 271. https://doi.org/10.1007/978-3-319-26024-2

[62] Singh, P., Dutta, M., Aggarwal, N. (2017). A review of task scheduling based on meta-heuristics approach in cloud computing. Knowl. Inf. Syst., 52(1): 1-51. https://doi.org/10.1007/s10115-017-1044-2

[63] Younis, M.T., Yang, S. (2018). Hybrid meta-heuristic algorithms for independent job scheduling in grid computing. Appl. Soft Comput., 72: 498-517. https://doi.org/10.1016/j.asoc.2018.05.032