A Detailed Classification of the Scheduling Algorithms in Fog Computing Environment, Challenges, and Future Directions

Hend Gamal El Din Hassan Ali, Imane Aly Saroit, Amira Mohamed Kotb
Faculty of Computers and Artificial Intelligence, Information Technology Dept., Cairo, Egypt

***Abstract*—Fog computing is considered as a distributed computing and storage resource. It is placed near to users rather than cloud computing to reduce latency in Internet of Things sensitive time applications. Several scheduling algorithms were proposed in fog environment to achieve better performance in terms of execution time, cost, latency and quality of services. In this research, a complete study of the different scheduling approaches in the fog computing is illustrated. Also, an operational classification of up-to-date algorithms is highlighted with detailed comparison of their performance metrics.**

***Keywords*—Cloud computing; fog computing; edge computing; Internet of Things; scheduling algorithms; directed acyclic graph**

I. Introduction

Cloud computing provides shared computing, storage resources and services to the users based on their demands using a variety of applications over the Internet. In cloud computing, all requests will be uploaded first to the centralized cloud servers to do the needed computations then the response will be sent back [1, 2]. Recently, Internet of Things (IoT) applications cover many areas in our life such as environment (smart home, smart city), society (health care, smart transportation), economy (smart factory)…,etc., [3]. As increasing usage of the IoT applications, massive amount of data flows in the used network, which leads to high traffic load [4]. Because of the massive data generated from the IoT devices and the long distance between them and the centralized cloud servers, this will lead to high overload over the cloud and high latency. This results many challenges for time-sensitive IoT applications like healthcare, intelligent traffic control and monitoring applications [5]. To overcome these challenges; other computing environment were introduced such as fog computing and edge computing [6].

Fog computing term was introduced by cisco systems in 2012 [7, 8]. Fog computing is considered as potential solution to latency produced in cloud computing. Fog computing can't be used instead of cloud computing but it is used with cloud computing to overcome the drawbacks of using only cloud [9]. Fog computing is a distributed computing paradigm that acts as an intermediate layer between the IoT sensors/devices and the cloud servers [10]. It offers computing, networking and storage resources that are near to the IoT devices rather than the cloud [11]. Fog computing provides less resources and storage capability than that of the cloud computing while it has fast response to requests than that of the cloud. The cloud computing is responsible for processing and executing if the fog computing layer can't execute the requests [9]. Due to the massive data generated by the IoT applications that need to be processed using the limited resources in any computing environment, scheduling process must be used to schedule the usage of these resources fairly between the user's requests (tasks) in cloud, fog or edge computing [12-14]. The contributions of this study are:

- First, provide a comprehensive study of various classification scheduling techniques in fog computing.
- Second, provide a detailed explanation of the recent algorithms related to each classification category.
- Third, compose a complete comparison between these algorithms by highlighting their strengths, limitations, performance metrics, submission task types, scheduling environment, solving technique, scheduling problem type and simulation tools.
- Finally, as a result of the applied comparison and subsequent of the statistical analysis, general direction for the future researches is proposed.

This study is organized as follows: In Section II, the detailed classifications of the scheduling techniques are illustrated with relevant scheduling algorithms associated with each category. A detailed comparison between the illustrated fog scheduling algorithms will be applied with discussion to statistical analysis obtained in Section III. Conclusion and summary are introduced in Section IV.

II. Classification of Scheduling Techniques

There are many types of classifications for the scheduling techniques used in the fog computing, as shown in Fig. 1. The selection of which used technique is based on the requirements of the system used in the fog computing. Classification can be done based on the type of the scheduling problem [7,15,16], based on decision time and amount of information needed before starting scheduling process like resource availability and task submission [17-20], based on the way of calculating the priority to tasks [21], based on the ability of interrupting the execution process [17,18], based on the used technique for solving the scheduling problem [19, 22-25], or based on the submission task types [20,22].

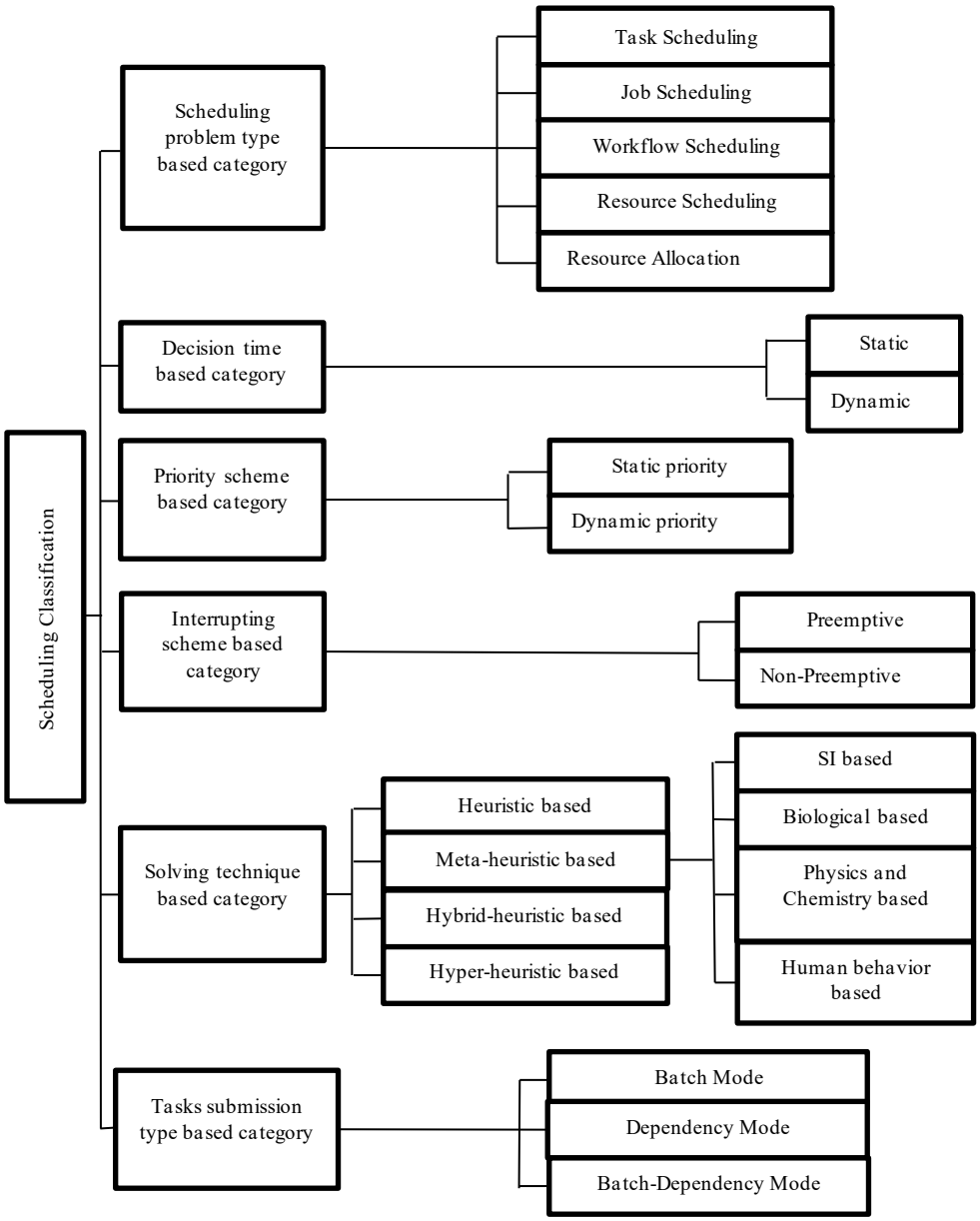

Fig. 1. Classifications of scheduling techniques.

A. Classification Based on Scheduling Problem Type

Fog scheduling protocols can be classified according to the type of the scheduling problem and their objective to five major types: task scheduling, job scheduling, workflow scheduling, resource scheduling and resource allocation [7].

- Task scheduling aims to assign tasks to the available resources (fog nodes) to satisfy the user's needs (QoS). In [9], task classification and virtual machine categorization method (TCVC) was proposed to schedule IoT healthcare tasks in the fog computing. TCVC is used in health care area to give urgent tasks high priority to be executed faster than those with high length. TCVC divides the tasks into three categories based on their importance. It also divides the VMs to three categories based on their capabilities. Each category of the VM services specific category of the tasks. Urgent tasks are executed in the fastest VM category. At each category of tasks, Max-Min algorithm is used to sort the tasks. In [26], prioritized task scheduling algorithm was proposed to calculate the priority to each incoming task based on its priority level, deadline and the available resources. Then it adds tasks in the priority queues (low, medium, high) based on their priority. The tasks with the highest priority (closer deadline) are processed first in the fog layer to meet the deadline. If the fog layer is busy, then the highest priority task is propagated to the cloud layer to achieve the deadline constrain.
- Job scheduling aims to assign group of jobs to minimum number of available resources (fog nodes) to get minimum CPU execution time. In [27], job scheduling algorithm with bag-of-tasks was proposed to

reduce the cost by utilizing all the available resources in the fog computing. All heterogeneous resources of fog computing (virtual machine or container, raspberry, smartphone or computer) are divided into sub-systems. Each subsystem can execute any job (independent and identically tasks). Each job is assigned to a sub-system based on the utilization calculation of the available resources and the given thresholds.

- Workflow scheduling aims to assign resources to workflows that include a set of jobs with different relationships (prerequisite, post-requisite). In [28], improved particle swarm optimization (IPSO) was proposed to schedule workflow in cloud and fog environment. Traditional particle swarm optimization (PSO) uses linear decreasing function to calculate inertia weight that is used in updating velocity and position of particle. This leads to instability between global and local search capabilities of particles. IPSO uses nonlinear decreasing function to calculate the inertia weight, which enhances the global and local search capability of particles. In [29], gravitational search algorithm (GSA) was proposed to get the optimal placement strategy to workflow (dependent processing elements). GSA follows the Newton's theory of gravity in which each particle attracts other particles with a force that is proportional to their masses and inversely proportional to the square of distance between them. So particles move toward other particles with heavy mass.

- Resource allocation aims to allocate the available resources to the clients based on their needs over the internet. In [30] the unit-slot optimization based on Lyapunov optimization technique was proposed to distribute all the incoming data into three-tier cloud of things (CoT) system; cloud, fog and edge. The Lyapunov optimization technique uses Lyapunov function with online systems to ensure stability, therefore unit-slot optimization is considered as online algorithm that doesn't need priori information before scheduling. A Lyapunov function is a nonnegative scalar measure of multi-dimensional vector of the system [31]. In unit-slot optimization, Lyapunov function is defined as the scalar measure of response time. In [32], the fog-enabled multi-tier operation scheduling (FEMOS) algorithm was proposed to model the delivery content of wireless network with heterogeneous resources. FEMOS is an online mode algorithm that uses Lyapunov optimization techniques to optimize task assignment (control tier) and resource allocation (access tier). In [33], dynamic task offloading framework was proposed to apply in dynamic fog environment uses deep Q-learning (DQL) that is a type of reinforcement learning (RL). The proposed framework models task allocation problem as Markov decision process (MDP).

- Resource scheduling aims to find the mapping between VM and the physical machines. In [34], online and utility-power efficient task scheduling algorithm was proposed to balance between the utility-power efficiency and average mean of queue backlogs performance in homogeneous fog networks. It offloads computation tasks from the wireless devices (WDs) to nearby fog nodes with available resources by modifying Lyapunov optimization techniques to maximize the utility-power while reducing the average mean of queue backlogs.

B. Classification Based on Decision Time

Fog scheduling protocols can be classified to two major types: static and dynamic, according to scheduling decision time and the amount of information needed about tasks and resources before starting scheduling process [17-19].

- Static scheduling needs all information about the available resources and tasks before starting scheduling as scheduling decisions are made before running time [20]. Static scheduling does not respond to any changes in the resources or tasks at runtime also there is no failure can occur during the scheduling. In [35], a profit-aware application placement policy was proposed to maximize provider profit (net profit and gross profit) by applying QoS requirements that present on deadline constrains. The proposed policy used integrated linear programming model (ILP) in fog and cloud computing environment. In [36], an improved jellyfish algorithm (IJFA) was proposed to minimize the task completion time, maximize resource utilization and improve the overall QoS. IJFA classifies tasks based on their priorities and deadline.

- Dynamic scheduling doesn't need prior information before scheduling as tasks are entered dynamically while the system is running. Scheduling decisions are made during running time. There is no need to wait for defining the incoming tasks and the available resources before starting the scheduling, also there is no guarantee to meet all constrains or tasks deadlines so the scheduler work hard to avoid failure [20]. In [37], dynamic scheduling algorithm was proposed to get the optimal responsive scheduling with distributed IoT architecture. Dynamic algorithm supports dynamic execution and re-scheduling based on fluctuation in the network parameters or QoS requirements. Dynamic scheduling uses two stages, the first stage aims to serve the tasks in local environment that leads to increase privacy and reduce traffic. For the remaining tasks, the second stage will be run using a user agent that sends the requests to all available fog and cloud nodes. In [38], the hybrid fog and cloud-aware heuristic for dynamic scheduling of multiple real-time IoT workflows (Hybrid-EDF) was proposed in two phases. The first phase is assigning a priority to each task based on its deadline by using earliest deadline first (EDF) policy. If there are many tasks with the same priority, the task with the highest average computational cost is selected first. The second phase is the VM selection for fog or cloud by using earliest estimated finish time (EFT). Hybrid-EDF finds gaps where the VM are idle and try to fill them with the existing tasks, in order to increase the utilization of system. In [39], dynamic task

allocation using a fuzzy logic enhanced (DTA-FLE) approach was proposed to classify tasks based on fuzzy logic among three layers low, high critical fog layer and cloud layer. Then DTA-FLE scheduled tasks in each layer to increase guarantee ratio and enhance performance.

C. Classification Based on Priority Scheme

Fog scheduling protocols can be classified according to the way of calculating the priority of tasks to two major types: static priority and dynamic priority [21].

- Static priority is assigned to tasks before execution time and cannot be changed during running time (priority depends on static attributes). In [40], prioritized incremental energy rate algorithm (PIER) was proposed to optimize the energy efficiency of the network. PIER presents scalable and robust priority based to schedule homogeneous offloading tasks with exponential distribution duration on fog nodes (FN). PIER minimizes the ratio of long-run average power consumption over the long-run average throughput with taking into consideration the availability of transmission channels and computing resources.
- Dynamic priority is assigned to tasks during running time and can be changed as it depending on the current state of system. Dynamic priority depends on dynamic attributes like latency or deadline as over time latency of unexecuted task increase while remaining deadline decrease. In [41], task priority dynamic implementation (TPDI) was proposed to reduce application loop lag and optimize performance of execution time, response time and minimize cost. TPDI applies scheduling process based on priority level and the classifications of the applications. TPDI assigns priorities to tasks on their arrival in a dynamic way. High priority tasks have minimum waiting time and execute earlier.

D. Classification Based on Interrupting Scheme

Fog scheduling protocols can be classified according to the ability of interrupting the execution process of the selected task to two major types: preemptive scheduling and non-preemptive scheduling [17, 18].

- Preemptive scheduling can stop the task that has been selected for execution before it is completed, which means it has the ability of interrupting the running task. In [42], fog computing scheduling algorithm was proposed for smart city namely weighted round-robin (WRR). The proposed algorithm used to handle huge number of requests that come from energy, water, transportation, housing and healthcare applications in smart city. The scheduling algorithm checks if the capability of the fog layer can execute the tasks or not. The task will be sent to the cloud layer using spanning-tree routing protocol (STP), if the resources in the fog layer not enough. Otherwise, WRR scheduling algorithm will be used to execute the tasks within the rotation cycle (RC) based on their priorities, and according to their available processing capability and remaining energy. WRR considers the task preemptive with no priority.
- Non-preemptive scheduling has to complete the task that has been selected for execution. The task cannot be stopped before it is finished. In [43], task scheduling in fog computing algorithm (TSFC) was proposed to schedule independent tasks based on relational table that includes results from I-Apriori algorithm between tasks and fog nodes. I-Apriori algorithm is proposed based on data mining Apriori algorithm. TSFC gets tasks with high priority from relational table and assigns them to fog nodes with minimum completion time. In [44], the priority and dependency-based DAG tasks offloading algorithm (PDAGTO) was proposed to schedule tasks with dependency constraints in heterogeneous fog nodes with multicore processors. PDAGTO assigns high priority to tasks with high computational overhead and high local energy consumption to meet the energy consumption requirement and minimizing the delay.

E. Classification Based on Solving Techniques

Fog scheduling protocols can be classified according to the way of solving the problem to four major types: heuristic, meta-heuristic, hybrid-heuristic and hyper-heuristic scheduling [19, 22-25].

- Heuristic scheduling is used to solve specific problem with known parameters defined by developers. Heuristic algorithms are simple and easy to implement but not guaranteed optimality [19, 22-24]. In [45], critical task first scheduler (CTFS) was proposed to schedule tasks based on its natural (critical or not critical) in healthcare scenario. CTFS gives critical tasks high priority unlike the other scheduling algorithms that give tasks with smaller MIPs size high priority than critical tasks with larger MIPs size. Therefore, incurs high latency to critical tasks with larger MIPs which is unacceptable in healthcare scenario. In [46], an efficient resource allocation and management strategies with energy efficiency (ERAM-EE) was proposed to assign IoT devices to fog nodes (FNs) through resource blocks (RBs). ERAM-EE constructs channel gain matrix and uses it to allocate RBs to IoT devices and FNs.
- Meta-heuristic scheduling is used to get general solution to the optimization problems [25]. Most of meta-heuristic techniques are natural-inspired. Meta-heuristic techniques have four categories: swarm intelligence (SI), biological based (bio-inspired), physics and chemistry based and human behavior based. SI are concerned with the behavior of multi agents that follow some simple rules like particular swarm optimization (PSO), ant colony optimization (ACO), bee life algorithm (BLA), artificial bee colony (ABC) algorithms, etc. SI is a subset from wider class called bio-inspired, so not all bio-inspired are SI. Genetic algorithm (GA), human inspired algorithm and whale optimization algorithm (WOA) are examples of the bio-

inspired based. Another type of meta-heuristic is inspired from the physics and chemistry based like harmony search and simulated annealing algorithm (SA). Human behavior based is observed from human societies like social emotional optimization algorithm (SEOA) and league championship algorithm [22,23,47]. In [48], the whale optimization algorithm (WOA) was inspired from the moving whales toward its preys. WOA is used to find the optimal energy management in smart power grids of plug-in hybrid electric vehicles (PHEVs). Data are encrypted using hash function in WOA to apply secure transaction between layers. In [49], opposition-based chaotic whale optimization algorithm (OppoCWOA) was proposed to improve the original version of WOA. OppoCWOA uses opposition-based learning and chaos theory with original WOA to extend the search space in both directions. Therefore OppoCWOA can avoid getting local optimal solution or premature convergence.

- Hybrid-heuristic scheduling combines two or more of algorithms to solve a predefined problem. Combined algorithms can be heuristic or meta-heuristic. Hybrid-heuristic algorithms gather the advantages and overcome the weakness of the combined algorithms. They achieve better performance than using one algorithm. But as they are integrated two or more algorithms at each iteration, it takes longer computation time [23,24]. In [50], a hybrid heuristic (HH) algorithm was proposed to combine IPSO algorithm with the improved ant colony optimization (IACO) algorithm, in order to schedule tasks of smart manufacturing system. First, the IPSO algorithm is used to get the optimal solution then forward its optimal solution as initial distribution to IACO algorithm. Then re-scheduling tasks using IACO to get the optimal result. HH algorithm combines the fast search from IPSO and the accuracy from IACO.

- Hyper-heuristic scheduling combines more than one algorithm like hybrid-heuristic but uses low-level heuristic (LLH) selector to decide which algorithms will be used at each iteration. Because only one algorithm will be used at each iteration, it achieves better performance than the hybrid-heuristic with less computation time [23,24]. In [23], hyper heuristic algorithm (HH) was proposed to combine four simple algorithms: genetic algorithm (GA), particular swarm optimization (PSO), ant colony optimization (ACO), and simulated annealing algorithm (SA). The HH is described in an intelligent surveillance case study with two phases; the training phase and the test phase. In the training phase, it applies the training workflow to the four techniques, then it calls the data center networks (DCNS) fog application to initialize the module mapping. And finally, it stores the best technique for each workflow in the dataset. In the test phase, it calculates the Euclidean distance between the new sample and the dataset. Then it chooses the best algorithm based on the lowest distance workflow. In [51], another hyper heuristic algorithm was proposed with the same idea of the HH algorithm but instead of calling the DCNS fog application, it calls the elderly human's activity detection (EHAD) in smart home case study, it also applies the security services with less overhead.

F. Classification Based on Submission Tasks Type

Fog scheduling protocols can be classified according to the way of acting with incoming tasks to three major types: batch, dependency, and batch-dependency. Fig. 2 shows different scheduling unit types used in scheduling algorithms [20].

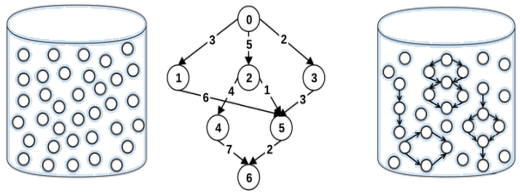

Fig. 2. Scheduling unit types.

- Batch algorithms work with independent tasks and cannot deal with dependent workflows. With this type, it is assumed that all the workflow is divided into independent parallel tasks first. In [52], time-cost aware scheduling (TCaS) algorithm was proposed to optimize the scheduling IoT bag of tasks problem based on an evolutionary genetic algorithm (GA). TCaS can balance between different user's requirements like fast execution or limit budget. In [53] drawer cosine optimization (DCO) was proposed for effective task offloading in fog computing. DCO combine between DA and SCA to apply multi-objectives such as makespan, cost, load, and energy. DCO is applied in master fog node that is responsible for scheduling the offloaded task requests. In [54], improved genetic algorithm for permutation-based optimization problems (IGA-POP) was proposed as a semi dynamic real time algorithm. IGA-POP achieves the minimum expected finish time and balance between the makespan, the total execution cost and deadline constraints.

- Dependency algorithms work with dependent workflows that are represented by directed acyclic graph (DAG). It doesn't have an efficient way to deal with parallel tasks [22]. In [55], cost-makespan aware scheduling heuristic algorithm (CMaS) was proposed in three phases. The first phase is assigning the priority to the dependent tasks in order to arrange them. Tasks are represented by directed acyclic graph (DAG) and ordered by upward ranking. The second phase is the selection phase that assigns each task to an appropriate processing node (cloud or fog) to achieve the optimal treatment between schedule length and cloud cost. The third phase is improving QoS while meeting the deadline constrains by reassigning the tasks in critical path to better processors to reduce the completion time. In [56], multi-agent system based genetic algorithm (MAS-GA) was proposed as a hybrid cloud-fog multi-agent approach applied with set of dependent IoT tasks.

MAS-GA has two phases. First, the MAS-GA divides the dependent workflow into sub-workflows by using the Mixed Min-Cut graph (MMCG) method. Then the agents set a contract that includes the deadline, the allowable budget and the required QoS metrics to each sub-workflow and sends the resource allocation request (RAR) to the closest fog agent. In the fog agent, it will compare RAR with the capability of the fog agent. If RAR is satisfied, then it uses genetic algorithm to find the nearest optimal allocation, else it will send the request to the cloud agent. In [57], electric earthworm optimization algorithm (EEOA) was applied in real time applications. EEOA combines the earthworm optimization algorithm (EOA) and the electric fish optimization algorithm (EFO) to improve searching explorations.

- Batch-dependency algorithms work first with dependency workflow and creates groups of independent tasks after orders them. Then they schedule the tasks in each group as batch algorithm [22]. In [58], an improved list-based task scheduling algorithm was operated in three stages. The first stage divides the tasks to levels, each level includes independent tasks. The second stage is assigning priority to the tasks at each level and then ordered these tasks based on their priorities. The last stage is assigning the resources to high priority tasks. In [59], hybrid list scheduling algorithm (HLSA) was proposed to apply with heterogeneous scheduling units with utilizing gaps in scheduling line. HLSA reduces makespan, total cost and latency to urgent tasks.

III. Comprehensive Comparison and Analysis

After presenting detailed classifications of the different approaches in the previous section, it is important to move on to a complete detailed comparison between the illustrated fog scheduling algorithms to evaluate their performance and determine which are most effective and suitable for diverse systems. This comparison is essential for all the future researches as it considered as a reference guideline to the previous algorithms and their required environment. Table I summaries the advantages, limitations, performance metrics and compared algorithms. Table II compared between illustrated algorithms in terms of computation environment, simulation tool, type of submission tasks, used technique and scheduling problem type.

Some statistical analysis that help for determining the future approach in fog scheduling algorithms are proposed based on a review of 34 research papers. Fig. 3 declared the performance metrics used to evaluate the illustrated scheduling algorithms. From that, we can observe that the most important metrics is cost by 47%. Cost can be computed based on computation process, communication process, usage of memory, bandwidth, energy, usage of cloud or services. Makespan, energy consumption, delay and execution time are considered the second most significant metric following cost by 41%, 41%, 35% and 32%, respectively.

Fig. 4 shows many simulation tools used to simulate fog scheduling algorithms. The most popular tool used is iFogSim by 27% followed by both cloudsim and Matlab with equal ratio by 15%. iFogSim is considered the extension of CloudSim by adding new functionality to it. Therefore, iFogSim and CloudSim represent 42% of the simulator tools used in the illustrated algorithms. However some research papers don't specify the used simulator tool by 15%.

Fig. 5 presents the main types of scheduling problems indicating the percentage of algorithms employed to each type. The task scheduling problem has the highest percentage by 56%, followed by 17% as resource allocation, 15% as workflow scheduling, 9% as job scheduling and 3% as resource scheduling.

Fig. 6 shows the percentage of each technique used for solving the illustrated scheduling problems. Heuristic-based scheduling algorithms have highest percentage by 59%, followed by 32% as meta-heuristic, 6% as hyper-heuristic and 3% as hybrid-heuristic.

Fig. 7 declared the percentage of task submission types used in illustrated algorithms. Therefore, 59% of scheduling algorithms can be applied to independent workflow, 29% can be applied to dependent workflow, 3% can be applied with both, 8% of research papers aren't mentioned the type of submission tasks.

Fig. 8 presents the usage percentage of each environment in illustrated algorithms. Thus, 50% of the scheduling algorithms depending on fog resources only compared with 47% depending on the extended resources from the cloud to meet some constrains like deadlines or the requirements of the tasks exceeded the fog resources while 3% used both fog and edge environment.

From this analysis, the general direction for the future research can be determined. First, it is necessary to specify the type of scheduling problem, the used technique, the type of workflow, simulation tool and environment. iFogSim is the most recommended simulation tool as it is an open source including CloudSim functionality with new features. Regarding the incoming workflow, some algorithms are applied to independent workflow without ability to deal with dependency constraints and others are applied to dependency workflow with the need to add assumptions to can apply with parallel tasks. So it is recommended to find approach that can be applied to all types of workflow without any restrictions on it. As for the used technique, hyper-heuristic technique needs more in-depth exploration as it combines more than one algorithm with less computation time and achieves better performance than using one algorithm. Hyper-heuristic technique gathers the advantages and overcome the weakness of the combined algorithms. Relative to the used environment, the system can either operate using existing resources in fog layer or can extend other resources from cloud layer with higher cost in case fog resources are insufficient. Regarding the scheduling problem type, task scheduling problem is the focal point of researches. As for the performance metrics, they are determined based on the requirements and objective of the system. Although this survey provides a comprehensive overview, it does not cover real-time constraints or benchmarking challenges.

TABLE I. A COMPARISON OF ILLUSTRATED SCHEDULING ALGORITHMS IN FOG COMPUTING

Protocol	Performance metrics	Advantage	Limitation	Comparison protocol used
DTA-FLE [39], 2025	Running time, Processing cost, Makespan, Delay, Energy consumption	Enhances scheduling efficiency. Improved Makespan. Minimize execution costs, energy and delays.	Computational process of fuzzy logic-based approach is complexity.	OEeRA, FBOT and DTA
DQL [33], 2025	Delay, Resource utilization, Energy consumption	Minimize latency and completion time. Efficient in real-time.	Not evaluated in real-world large-scale deployments.	PPO, PSO and Synergistic AI Framework
DCO [53], 2025	Execution time, Load balancing, Makespan, Memory, Energy consumption	Minimize latency, energy consumption and total offloading time. Meet deadline. Efficiently offloads prioritized tasks to the proper nodes.	Not considered the network conditions and resource availability to make real-time decisions.	SACO, ACO, JTORA, Multi-objective fuzzy approach, Improved contract net with BAS and DCTO
HLSA [59],2025	Makespan, Latency, Total cost (communication and computation cost).	Reduce latency, cost and Makespan. Apply with heterogeneous scheduling units. Utilizing gaps in scheduling line.	Can't apply with dynamic environment.	Heterogeneous Earliest Finish Time (HEFT), Min-Min, TS-QoS, Improved list task scheduling algorithm
Online task scheduling alg. [34], 2024	Utility-power efficiency, Average mean of queue backlogs	Maximize utility-power efficiency. Minimizing average of queue backlogs.	Not compared with other algorithms.	Compared with it-self using different parameters
ERAM-EE [46], 2024	Response time, Processing time, Energy efficiency	Improves the energy efficiency, the response and process time. Reduces the volume of data processing and storage in the cloud.	Can't apply with dynamic and heterogeneous natural. It has constraints with CPU, memory and processing power.	RR-EPA, OR-EPA, and EE-CN
TPDI [41], 2023	Tasks: Distribution time, Waiting time, Execution time. Applications: Execution time, Network usage, Energy consumption	Apply with heterogeneous environment to reduce response time, execution time and cost.	Not considered the other QoS parameters.	FCFS and SJF
EEOA [57], 2023	Makespan, Total cost (communication and computation cost), Energy consumption	Enhance convergence speed and searching explorations. Takes the QoS and energy requirements into account. Validate for sensitive application.	Can't predict upcoming workloads and cannot decide whether or not to offload the task.	CSPSO, CSDEO, H-DEWOA, and BLEMO
PIER [40], 2022	Energy efficiency	Robustness to different distribution of task duration. Improve energy efficiency.	Not considered the other QoS parameters.	PTR and PLPC
CTFS [45], 2022	Latency, Network utilization, Energy consumption	Minimize latency for sensitive time applications, energy consumption, response time and network utilization.	Can't apply with heterogeneous and dynamic environment.	FCFS, SJF and cloud-only approaches
IGA-POP [54], 2022	Makespan, Delay time, Failure rate, Execution cost	Provide best permutation for real-time tasks. Minimize finish time, average delay time and failure rate. Balance between Makespan and execution cost. Meeting deadline constraints.	Not apply in dynamic environment. Not adapted to address the workflow scheduling problem.	First fit, Best fit, Bees life and GA
IJFA [36], 2022	Makespan, Completion Time	Minimize the task completion time. Maximize resource utilization. Improve the overall QoS.	Not considered the response time and energy in fog-cloud architecture.	JS
Dynamic scheduling alg. [37], 2022	Latency, Throughput	Apply dynamic re-scheduling based on the changeable QoS parameters. Improve throughput and latency.	Rescheduling always depend on user agent.	Kube-scheduler and NAS
PDAGTO [44], 2021	Delay, Offloading ratio, Energy consumption	Reduce delay in fog node and total energy consumption of the device. Increase CPU utilization.	PDAGTO is batch processing and offline algorithm.	DAG offloading algorithm (DAGOA)
WRR [42], 2021	Delay, Complexity, Throughput	Improve performance in throughput, latency and fairness.	Not apply in dynamic environment.	FogFlow approach
MAS-GA [56], 2021	Response time, Cost, Reliability, Availability, Execution time, Makespan	Maximize the usage of fog to minimize response time. Use cloud computing resources with minimum cost.	Not considered the interactions between agents.	Compared with itself in cloud only and in fog only
Improved list task scheduling alg. [58], 2021	Speed up, Makespan, Cost (computation time of tasks in critical path), Scheduling length ratio, Running time	Minimize Makespan and overall cost. Balance between local and global optimal approach. Restricted cross-over between processors.	Not apply in dynamic environment.	PEFT, HEFT, MOPT, and SDBATS

OppoCWOA [49], 2021	Execution time, Energy consumption	Achieve better performance time-energy efficient.	Process of learn and update weights take long time. Not apply with dependent workflow.	WOA, PSO, GA and ABC
WOA [48], 2021	Cost of energy	Decreasing the loss of power. The transactions of data are secured among the layers.	Not concerned with time.	PSO and GA
Profit-aware application placement [35], 2020	Waiting time, Profit(Gross, Net), Percentage of QoS satisfied	Maximize profit without violation deadline constraints. Minimize waiting time.	Not concerned with security aspects.	Heuristic profit-aware, Deadline-prioritized, Completion time-prioritized, Revenue-prioritized
TCVC [9], 2019	Waiting time, Finished time, Execution time	Achieve fairness between tasks and load balancing between VM. Minimize execution time and latency to sensitive tasks.	VMs capability should be proportional with the important of tasks.	Max-Min, FCFS and SJF
Scheduling alg. with bag of tasks [27], 2019	Resource utilization, Response time, Delay, Throughput, Cost (energy consumption and using cloud cost)	Reduce cost and decrease execution time. Adapting with resources in changing environment.	The overall energy consumption of the system not examined.	Weighted Round Robin
FEMOS [32], 2019	Delay, Fairness, Throughput	Balance between throughput and delay. Confirm the benefit of centralized assignment and dynamic online bandwidth allocation.	Not examined the proactive FAN assignment and resource management problem.	SBC and SLQ
IPSO [28], 2019	Makespan, Cost (computation cost)	Minimize completion time of workflow compared with PSO with similar cost.	Not apply multiple objectives optimization.	PSO
Hybrid-EDF [38], 2019	Cost (using cloud and communication time), Deadline	Meets deadline constraints. Utilizing possible gaps in the schedule of fog and cloud VMs.	Reserved dedicated hosts in cloud layer that lead to increase cost.	Fog-EDF
GSA [29], 2019	Response time	Minimize response time with cost limitations.	Concerned only with response time.	PSO
TCaS [52], 2019	Makespan, Cost (CPU, memory, bandwidth usage)	Can balance between minimum Makespan and execution cost. Satisfy user's requirement.	Not concerned to time, energy consumption and cost or constraints of budget, deadline and resource limitation.	BLA, MPSO and RR
HH [50], 2019	Makespan, Reliability, Energy consumption	Achieve better performance in Makespan, reliability and energy consumption.	Not apply task clustering.	IPSO, IACO and RR
Prioritized task scheduling alg. [26], 2018	Response time, Cost (Computation and communication time)	Some alg. with proposed priority version improved response time and other improved cost so can choose the suitable algorithm based on the system requirements.	There is a static threshold for rejecting task.	ORT, RD and ERA compared with ORT, RD and ERA with proposed priority version
TSFC [43], 2018	Waiting time, Execution time	Minimize the total execution time of tasks and average waiting time.	Not concerned with bandwidth between processors and multilayer task scheduling.	MCT, MET and Min-Min
HH [23], 2017	Execution time, Cost (bandwidth, storage processor, memory), Energy consumption, Network usage	Reduce energy consumption, cost and execution time.	Not address virtual machine's migration, resource provisioning.	GA, PSO, ACO and SA
HH [51], 2017	Execution time, Cost, Security (integrity, authentication, confidentiality), Network usage, Energy consumption	Decrease the risk in security services. Reduce energy consumption, cost and execution time.	Not address virtual machine's placement, resource provisioning.	GA, PSO, ACO and SA
CMaS [55], 2017	Makespan, Deadline, Cost	Balancing between fast executions to apply deadline constraints and reducing cost.	Not deploy in real-world systems.	GfC, HEFT and CCSH
Unit-slot optimization [30], 2016	Response time, Cost (communication and computation time)	Achieve good balancing between cost and response time.	Uncertain communication between three tiers CoT.	Fog-First and Cloud-First

TABLE II. A DETAILED COMPARISON OF ILLUSTRATED SCHEDULING ALGORITHMS IN FOG COMPUTING

Protocol	Environment	Simulation Tool	Task Type	Techniques	Scheduling Problem

DTA-FLE [39], 2025	Fog/cloud	iFogSim	Independent	Heuristic	Resource allocation
DQL [33], 2025	Fog	Undeclared	Independent	Heuristic	Resource allocation
DCO [53], 2025	Fog	Python	Independent	Meta-heuristic	Task scheduling
HLSA [59],2025	Fog	Java	Independent / Dependent	Heuristic	Task scheduling
Online task scheduling alg. [34], 2024	Fog	Undeclared	Independent	Heuristic	Resource scheduling
ERAM-EE [46], 2024	Fog	MATLAB	Independent	Heuristic	Task scheduling
TPDI [41], 2023	Fog	iFogSim	Independent	Heuristic	Resource allocation
EEOA [57], 2023	Fog/cloud	CloudSim	Dependent	Meta-heuristic	Task scheduling
PIER [40], 2022	Fog	Custom simulation	Independent	Heuristic	Task scheduling
CTFS [45], 2022	Fog	iFogSim	Dependent	Heuristic	Task scheduling
IGA-POP [54], 2022	Fog/cloud	MATLAB	Independent	Meta-heuristic	Task scheduling
IJFA [36], 2022	Fog/cloud	MATLAB	Independent	Meta-heuristic	Task scheduling
Dynamic scheduling alg. [37], 2022	Fog/cloud	IMUNES	Independent	Heuristic	Task scheduling
PDAGTO [44], 2021	Fog/Edge	Undeclared	Dependent	Heuristic	Task scheduling
WRR [42], 2021	Fog/cloud	NS 3.26 and iFogSim	Independent	Heuristic	Task scheduling
MAS-GA [56], 2021	Fog/cloud	WorkflowSim extends from CloudSim	Dependent	Meta-heuristic	Workflow scheduling
Improved list task scheduling alg. [58], 2021	Fog	iFogSim	Dependent	Heuristic	Task scheduling
OppoCWOA [49], 2021	Fog	Python with PyCharm editor	Independent	Meta-heuristic	Task scheduling
WOA [48], 2021	Fog/cloud	Monte Carlo Simulation	Dependent	Meta-heuristic	Workflow scheduling
Profit-aware application placement [35], 2020	Fog/cloud	iFogSim	Independent	Heuristic	Resource allocation
TCVC [9], 2019	Fog	CloudSim	Independent	Heuristic	Task scheduling
Scheduling alg. with bag of tasks [27], 2019	Fog	C	Independent	Heuristic	Job scheduling
FEMOS [32], 2019	Fog	Undeclared	Independent	Heuristic	Resource allocation
IPSO [28], 2019	Fog/cloud	MATLAB	Dependent	Meta-heuristic	Workflow scheduling
Hybrid-EDF [38], 2019	Fog/cloud	C++	Dependent	Heuristic	Workflow scheduling
GSA [29], 2019	Fog/cloud	Undeclared	Dependent	Meta-heuristic	Workflow scheduling
TCaS [52], 2019	Fog/cloud	iFogSim	Independent	Meta-heuristic	Task scheduling
HH [50], 2019	Fog	MATLAB	Independent	Hybrid heuristic	Task scheduling
Prioritized task scheduling alg. [26], 2018	Fog/cloud	Cloud Analyst built on top of CloudSim	Independent	Heuristic	Task scheduling
TSFC [43], 2018	Fog	SimGrid	Independent	Heuristic	Task scheduling
HH [23], 2017	Fog	iFogSim	Not mention	Hyper heuristic	Job scheduling
HH [51], 2017	Fog	iFogSim	Not mention	Hyper heuristic	Job scheduling
CMaS [55], 2017	Fog/cloud	CloudSim	Dependent	Heuristic	Task scheduling
Unit-slot optimization [30], 2016	Fog/cloud	SimPy	Not mention	Meta-heuristic	Resource allocation

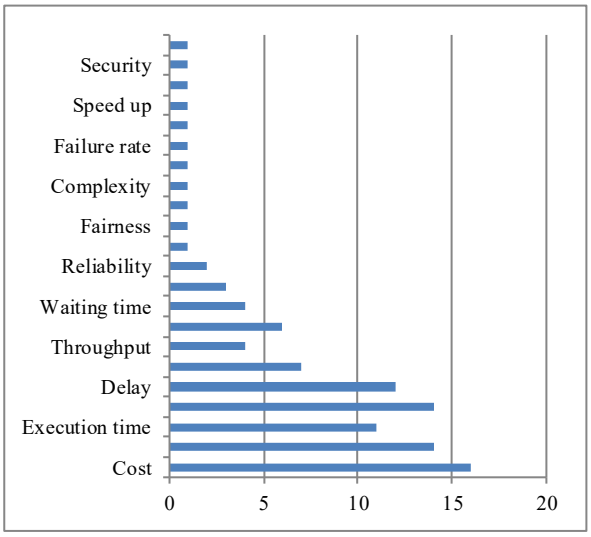

Fig. 3. Usage of scheduling metrics.

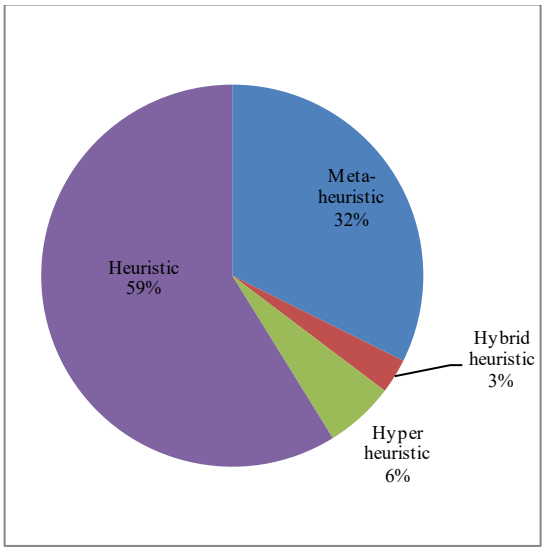

Fig. 6. Percentage of scheduling technique.

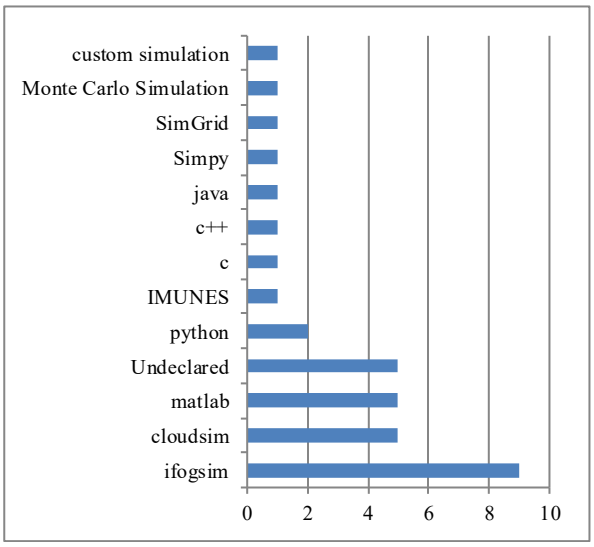

Fig. 4. Usage of simulation tools.

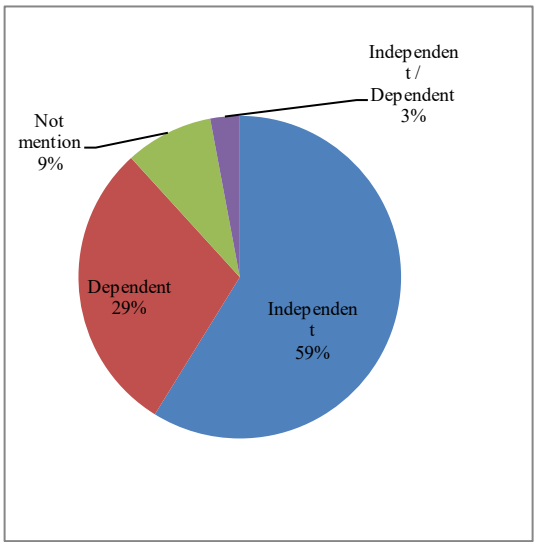

Fig. 7. Percentage of task submission types.

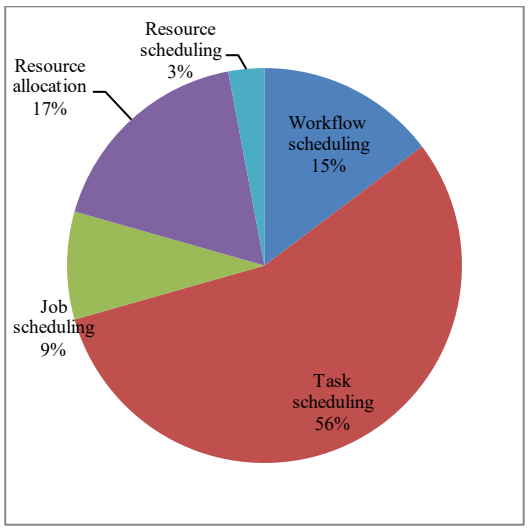

Fig. 5. Percentage of scheduling problem types.

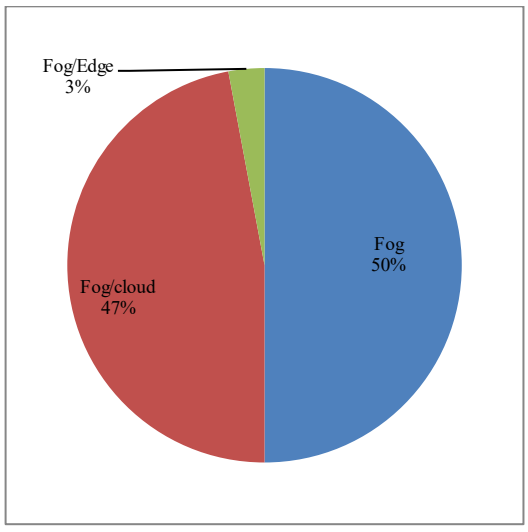

Fig. 8. Percentage of computing environment.

IV. CONCLUSION

This study illustrated a detailed classification for the recent scheduling algorithms in fog computing. For each category in the classification, identified algorithms that belong to it. A complete comparison is adopted between the illustrated algorithms to highlight their advantages and challenges. This comparison helps researchers to find the appropriate technique to specific problem. Furthermore, statistical analysis is applied and clarified the main research direction.

REFERENCES

[1] N. Rajeswari, "Overview of Cloud Computing and Its Types," SSRN Electronic Journal, vol. 6, pp. 61–67, 2019. Available: http://www.jetir.org/papers/JETIRAT06008.pdf.

[2] A. Bhutto, A. A. Chandio, K. K. Luhano, and I. A. Korejo, "Analysis of Energy and Network Cost Effectiveness of Scheduling Strategies in Datacentre," Cybernetics and Information Technologies, vol. 23, no. 3, pp. 56–69, 2023. doi:10.2478/cait-2023-0024.

[3] F. Khodadadi, A. Vahid, and R. Buyya, "Internet of Things: An Overview," arXiv preprint arXiv:1703.06409, 2017. doi:10.48550/arXiv.1703.06409.

[4] E. S. Lowe, T. Tiropanis, and W. Hall, "Analytics for the Internet of Things: A Survey," ACM Computing Surveys, 2018. doi:10.1145/3204947.

[5] M. Dang, J. Piran, D. Han, K. Min, and H. Moon, "A Survey on Internet of Things and Cloud Computing for Healthcare," Electronics, vol. 8, no. 7, 2019. doi:10.3390/electronics8070768.

[6] R. Mahmud, R. Kotagiri, and R. Buyya, "Fog Computing: A Taxonomy, Survey and Future Directions," Springer, Singapore, pp. 103–130, 2018. doi:10.1007/978-981-10-5861-5_5.

[7] K. Matrouk and K. Alatoun, "Scheduling Algorithms in Fog Computing: A Survey," International Journal of Networked and Distributed Computing, vol. 9, pp. 59–74, 2021. doi:10.2991/ijndc.k.210111.001.

[8] Y. I. Alzoubi and A. Aljaafreh, "Blockchain-Fog Computing Integration Applications: A Systematic Review," Cybernetics and Information Technologies, vol. 23, no. 1, 2023. doi:10.2478/cait-2023-0001.

[9] T. Aladwani, "Scheduling IoT Healthcare Tasks in Fog Computing Based on Their Importance," Procedia Computer Science, 2019, pp. 560–569. doi:10.1016/j.procs.2019.12.138.

[10] W. Yu et al., "A Survey on Edge Computing for the Internet of Things," IEEE Access, vol. 6, 2018, pp. 6900–6919. doi:10.1109/ACCESS.2017.2778504.

[11] A. Monika and M. Kinranangia, "The Role of Cloud and Fog Computing in IoT," International Journal of Scientific Research and Review, vol. 7, pp. 607–612, 2019. doi:10.32628/IJSRR1907001.

[12] K. Dolui and S. K. Datta, "Comparison of Edge Computing Implementations: Fog Computing, Cloudlet and Mobile Edge Computing," IEEE Global Internet of Things Summit (GIoTS), 2017. doi:10.1109/GIOTS.2017.8016213.

[13] M. E., "A Survey of Various Scheduling Algorithm in Cloud Computing Environment," Journal of Emerging Technologies and Innovative Research (JETIR), vol. 2, pp. 131–135, 2013. doi:10.15623/IJRET.2013.0202008.

[14] K. Bhargavi, S. Babu, and S. G. Shiva, "Type-2-Soft-Set Based Uncertainty Aware Task Offloading Framework for Fog Computing Using Apprenticeship Learning," Cybernetics and Information Technologies, vol. 23, no. 1, pp. 38–58, 2023. doi:10.2478/cait-2023-0002.

[15] M. Hosseinzadeh et al., "Task Scheduling Mechanisms for Fog Computing: A Systematic Survey," IEEE Access, 2023. doi:10.1109/ACCESS.2023.3277826.

[16] S. Bansal, H. Aggarwal, and M. Aggarwal, "A Systematic Review of Task Scheduling Approaches in Fog Computing," Transactions on Emerging Telecommunication Technologies, 2022. doi:10.1002/ett.4523.

[17] G. Amalarethinam and P. Muthulakshmi, "An Overview of the Scheduling Policies and Algorithms in Grid Computing," International Journal of Research and Reviews in Computer Science (IJRRCS), vol. 2, no. 2, 2011. doi:10.1109/ICCT.2008.371.

[18] H. Goel and N. Chamoli, "Job Scheduling Algorithms in Cloud Computing: A Survey," International Journal of Computer Applications, vol. 95, no. 23, pp. 19–22, 2014. doi:10.5120/16735-6981.

[19] M. Reza et al., "A Task Scheduling Approaches in Fog Computing: A Systematic Review," International Journal of Communication Systemic, vol. 33, 2020. doi:10.1002/dac.4583.

[20] P. Varshney and Y. Simmhan, "Characterizing Application Scheduling on Edge, Fog and Cloud Computing Resources," Software Practice and Experience, vol. 50, 2019. doi:10.1002/spe.2699.

[21] R. Patel and H. Mer, "A Survey of Various QoS-Based Task Scheduling Algorithm in Cloud Computing Environment," International Journal of Cloud Computing and Management (IJCCM), vol. 4, 2013. doi:10.15660/ijcsm/2013/112/13110.

[22] J. Yu, R. Buyya, and K. Ramamohanarao, "Workflow Scheduling Algorithms for Grid Computing, Metaheuristics for Scheduling in Distributed Computing Environments," Springer, vol. 146, pp. 173–214, 2008. doi:10.1007/978-3-540-69277-5_7.

[23] S. Kabirzadeh, D. Rahbari, and M. Nickray, "A Hyper Heuristic Algorithm for Scheduling of Fog Networks," IEEE Open Innovations Association FRUCT Conference, vol. 562, Finland, 2017, pp. 148–155. doi:10.23919/FRUCT.2017.8250177.

[24] C. Tsai et al., "A Hyper-Heuristic Scheduling Algorithm for Cloud: Brief Review of Nature-Inspired Algorithms for Optimization," IEEE Transactions on Cloud Computing, vol. 2, 2014, pp. 236–250. doi:10.1109/TCC.2014.2315797.

[25] I. Fister et al., "Brief Review of Nature-Inspired Algorithms for Optimization," Elektrotehniski Vestnik/Electrotechnical Review, vol. 3, 2013. doi:10.48550/arXiv.1307.4186.

[26] T. Choudhari, "Prioritized Task Scheduling in Fog Computing," Master Project in SJSU ScholarWorks, 2018, pp. 1–8. doi:10.1145/3190645.3190699.

[27] D. Tychalas and H. Karatza, "A Scheduling Algorithm for a Fog Computing System with Bag-of-Tasks Jobs: Simulation and Performance Evaluation," Simulation Modelling Practice and Theory, vol. 98, 2019. doi:10.1016/j.simpat.2019.101982.

[28] R. Xu et al., "Improved Particle Swarm Optimization-Based Workflow Scheduling in Cloud-Fog Environment," Springer Nature Switzerland, vol. 342, pp. 337–347, 2019. doi:10.1007/978-3-030-11641-5_27.

[29] A. Karamoozian, A. Hafid, and E. Aboulhamid, "On the Fog-Cloud Cooperation: How Fog Computing Can Address Latency Concerns of IoT Applications," IEEE International Conference on Fog and Mobile Edge Computing (FMEC), 2019, pp. 166–172. doi:10.1109/FMEC.2019.8795320.

[30] B. Polyak and P. Shcherbakov, "Lyapunov Functions: An Optimization Theory Perspective," IFAC Journal, vol. 50, 2017, pp. 7456–7461. doi:10.1016/j.ifacol.2017.08.1513.

[31] Y. Nan et al., "Cost Effective Processing for Delay Sensitive Applications in Cloud of Things Systems," IEEE International Symposium on Network Computing and Applications, 2016, pp. 162–169. doi:10.1109/NCA.2016.7778612.

[32] S. Zhao et al., "FEMOS: Fog-Enabled Multi-Tier Operations Scheduling in Dynamic Wireless Networks," IEEE Global Communications Conference (Globecom), Singapore, 2019. doi:10.1109/JIOT.2018.2808280.

[33] H. M. Abdelghany, "Dynamic Resource Management and Task Offloading Framework for Fog Computing," Grid Computing Journal, vol. 23, 2025. doi:10.1007/s10723-025-09804-7.

[34] F. Ebadi and V. Shah-Mansouri, "Online and Utility-Power Efficient Task Scheduling in Homogeneous Fog Networks," IEEE Networking and Internet Architecture, 2024. doi:10.48550/arXiv.2409.18675.

[35] R. Mahmud et al., "Profit-Aware Application Placement for Integrated Fog-Cloud Computing Environments," Parallel Distributed Computing Journal, vol. 153, 2020. doi:10.1016/j.jpdc.2019.10.001.

[36] N. Jangu and Z. Raza, "Improved Jellyfish Algorithm-Based Multi-Aspect Task Scheduling Model for IoT Tasks over Fog Integrated Cloud Environment," Journal of Cloud Computing, 2011. doi:10.1186/s13677-022-00376-5.

[37] P. Krivic et al., "Dynamic Scheduling of Contextually Categorised Internet of Things Services in Fog Computing Environment," Journal of Sensors, vol. 22, 2022. doi:10.3390/s22020465.

[38] G. L. Stavrinides and H. D. Karatza, "A Hybrid Approach to Scheduling Real-Time IoT Workflows in Fog and Cloud Environments," Springer Multimedia Tools and Applications, vol. 78, 2019. doi:10.1007/s11042-018-7051-9.

[39] W. Jin and A. Rezaeipanah, "Dynamic Task Allocation in Fog Computing Using Enhanced Fuzzy Logic Approaches," Scientific Reports, vol. 15, 2025. doi:10.1038/s41598-025-03621-4.

[40] J. Yin et al., "Energy Efficient Priority-Based Task Scheduling for Computation Offloading in Fog Computing," IEEE Transactions on Network and Service Management, 2022, pp. 564–577. doi:10.1007/978-3-030-95384-3_35.

[41] A. Shoker et al., "Resource Allocation Strategy in Fog Computing: Task Scheduling in Fog Computing Systems," Journal of Communication Sciences and Information Technology (JCSIT), vol. 1, 2023, pp. 1–11. doi:10.21608/jcsit.2023.306757.

[42] A. Mohammad et al., "Fog Computing Scheduling Algorithm for Smart City," International Journal of Electrical and Computer Engineering (IJECE), vol. 11, 2021, pp. 2219–2228. doi:10.11591/ijece.v11i3.pp2219-2228.

[43] L. Liu et al., "A Task Scheduling Algorithm Based on Classification Mining in Fog Computing Environment," Wireless Communications and Mobile Computing, 2018. doi:10.1155/2018/2102348.

[44] X. Fu et al., "Priority and Dependency-Based DAG Tasks Offloading in Fog/Edge Collaborative," IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD), 2021, pp. 440–445. doi:10.1109/CSCWD49262.2021.9437784.

[45] A. Khan et al., "Effective Task Scheduling in Critical Fog Applications," Scientific Programming, 2022. doi:10.1155/2022/9208066.

[46] P. Periasamy et al., "ERAM-EE: Efficient Resource Allocation and Management Strategies with Energy Efficiency under Fog–Internet of Things Environments," Connection Science, vol. 36, 2024. doi:10.1080/09540091.2024.2350755.

[47] Y. Xu, Z. Cui, and J. Zeng, "Social Emotional Optimization Algorithm for Nonlinear Constrained Optimization Problems," International Conference on Swarm, Evolutionary, and Memetic Computing (SEMCCO), 2010. doi:10.1007/978-3-642-17563-3_68.

[48] Y. Lin et al., "A DAG Based Cloud-Fog Layer Architecture for Distributed Energy Management in Smart Power Grids in the Presence of PHEVs," Journal of Sustainable Cities and Society, vol. 75, 2021. doi:10.1016/j.scs.2021.103335.

[49] Z. Movahedi, B. Defude, and A. Mohammad, "An Efficient Population Based Multi-Objective Task Scheduling Approach in Fog Computing Systems," Journal of Cloud Computing, vol. 10, 2021. doi:10.1186/s13677-021-00264-4.

[50] J. Wang and D. Li, "Task Scheduling Based on a Hybrid Heuristic Algorithm for Smart Production Line with Fog Computing," Journal of Sensors, vol. 19, 2019. doi:10.3390/s19051023.

[51] D. Rahbari, S. Kabirzadeh, and M. Nickray, "A Security Aware Scheduling in Fog Computing by Hyper Heuristic Algorithm," IEEE Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS), 2017, pp. 87–92. doi:10.1109/ICSPIS.2017.8311595.

[52] B. Minh et al., "Evolutionary Algorithms to Optimize Task Scheduling Problem for the IoT Based Bag-of-Tasks Application in Cloud–Fog Computing Environment," Journal of Applied Sciences, 2019. doi:10.3390/app9091730.

[53] B. Ameena and L. Ramasamy, "Drawer Cosine Optimization Enabled Task Offloading in Fog Computing," Expert Systems with Applications, vol. 259, 2025. doi:10.1016/j.eswa.2024.125212.

[54] A. S. Abohamama, A. El-Ghamry, and E. Hamouda, "Real-Time Task Scheduling Algorithm for IoT-Based Applications in the Cloud–Fog Environment," Journal of Network and Systems Management, vol. 30, 2022. doi:10.1007/s10922-022-09664-6.

[55] X. Pham et al., "A Cost and Performance Effective Approach for Task Scheduling Based on Collaboration Between Cloud and Fog Computing," International Journal of Distributed Sensor Networks, vol. 13, 2017. doi:10.1177/1550147717742073.

[56] M. Mokni et al., "Cooperative Agents Based Approach for Workflow Scheduling on Fog Cloud Computing," Springer Journal of Ambient Intelligence and Humanized Computing, vol. 13, 2021. doi:10.1007/s12652-021-03187-9.

[57] M. S. Kumar and G. R. Karri, "EEOA: Cost and Energy Efficient Task Scheduling in a Cloud-Fog Framework," Journal of Sensors, vol. 23, 2023. doi:10.3390/s23052445.

[58] R. Madhura et al., "An Improved List Based Task Scheduling Algorithm for Fog Computing Environment," Computing, vol. 103, 2021. doi:10.1007/s00607-021-00935-9.

[59] H. G. E. H. Ali, I. A. Saroit, and A. M. Kotb, "HLSA – A New Hybrid List Scheduling Algorithm for Fog Computing," Cybernetics and Information Technologies, vol. 25, no. 3, pp. 123–141, 2025. doi:10.2478/cait-2025-0026.