# Resource Allocation and Task Scheduling in Fog Computing and Internet of Everything Environments: A Taxonomy, Review, and Future Directions

Bushra Jamil, Humaira Ijaz, Mohammad Shojafar, *Senior Member, IEEE*
Kashif Munir and Rajkumar Buyya *Fellow, IEEE*

***Abstract*—Internet of Everything (IoE) paradigm is being rapidly adopted in developing applications for different domains like smart agriculture, smart city, big data streaming, etc. These IoE applications are leveraging cloud computing resources for execution. Fog computing, which emerged as an extension of cloud computing, supports mobility, heterogeneity, geographical distribution, context awareness, and services like storage, processing, networking, and analytics on nearby fog nodes. The resource-limited, heterogeneous, dynamic, and uncertain fog environment makes task scheduling a great challenge that needs to be investigated. This paper is motivated by this consideration and presents a systematic, comprehensive and detailed comparative study by discussing the merits and demerits of different scheduling algorithms, focused optimization metrics, and evaluation tools in the fog computing and IoE environment. The goal of this survey paper is fivefold. First, we review the fog computing and IoE paradigms. Second, we delineate the optimization metric engaged with fog computing and IoE environment. Third, we highlight the heuristic, metaheuristic scheduling algorithms dealing with fog computing and IoE environment paradigms by leveraging some examples. Fourth, we rationalize the scheduling algorithms and point out the lesson learned from the survey. Fifth, we discuss the open issues and future research directions to improve scheduling in fog computing and IoE environment.**

***Index Terms*—Cloud Computing, Fog Computing, Internet of Things (IoT), Internet of Everything (IoE), Resource Allocation, Task Scheduling.**

## I. Introduction

THE Internet of Everything (IoE) paradigm is based on the convergence of the digital and physical world to make this world smarter with intelligence, cognition, and connectivity. IoE is the system that interconnects billions of heterogeneous physical devices, computing elements, objects, animals, and humans that can setup, share, and self-organize their limited resources to achieve a system-wide goal [1]. The main objective of IoE networks is to enhance the performance of underlying Internet of Things (IoT) physical infrastructures by providing services to humans [2].

The IoT paradigm aims to make things smart without human intervention, by incorporating in them the ability to capture, send data over the network, and respond according to given commands [3]. Any entity such as sensors, actuators, and Radio Frequency Identification (RFID) devices with an Internet Protocol (IP) address, along with the ability to capture and transmit data over the network, can be a thing [4]. These things are the only pillar of the IoT, so communication is always between these things and machines through various protocols. The terms IoT and IoE are interchangeably used. These IoE devices are resource and energy constrained and cannot efficiently process data generated by various IoE applications within a fixed time frame. Therefore, to host these IoE applications, cloud computing has been the most commonly used distributed computing paradigm in recent years due to its capabilities like unlimited storage and processing power, service-oriented architecture, high performance, reliability, and on-demand access to the shared pool of configurable resources [5], [6]. Currently, a centralized Cloud-centric Internet of Things (CIoT) architecture is used to process the data generated by these IoE devices-based IoE applications [7].

The recent advancements in IoT and computing technologies, use of ubiquitous and pervasive computing for provision of computing services all the time and at any location, and other enabling communication technologies like 4G, 5G and Wireless Sensor Network (WSN), have resulted in a rapidly increasing number of IoE devices and applications. According to an estimate, active IoE devices will reach up to 1.2 trillion by 2030 [8]. These IoE devices generate an enormous amount of different-nature data used by different applications. The cloud data centers alone will be unable to efficiently process this vast amount of data generated by these devices. Furthermore, the cloud data centres are multi-hop away from the end-user. Therefore, transferring data from end-user to geographically-distant cloud results in long latency and network congestion that are not acceptable for time-sensitive applications like smart healthcare, connected vehicles, remote patient monitoring, etc [9]. Therefore, in the near future, the traditional centralized CIoT architecture will not be able to address these challenges of bandwidth consumption, long delay, network congestion, and security.

To handle these issues, several approaches like mist computing, mobile cloud computing, multi-access edge computing, volunteer computing, cloudlet computing, and fog computing

B. Jamil and H. Ijaz are with the Department of CS & IT, University of Sargodha, Sargodha, Pakistan.
E-mail: {bushra.jamil,humaira.bilalrasul}@uos.edu.pk
M. Shojafar is with 5G & 6G Innovation Center, Institute for Communication Systems, University of Surrey, UK, Guildford, UK.
E-mail: m.shojafar@surrey.ac.uk
K. Munir is with Department of Computer Science, National University of Computer and Emerging Sciences, Islamabad, Pakistan. E-mail: kashif.munir@nu.edu.pk
Rajkumar Buyya is with the Cloud Computing and Distributed Systems (CLOUDS) Lab, School of Computing and Information Systems, University of Melbourne, Australia.
E-mail: rbuyya@unimelb.edu.au.
(Corresponding author: Humaira Ijaz)

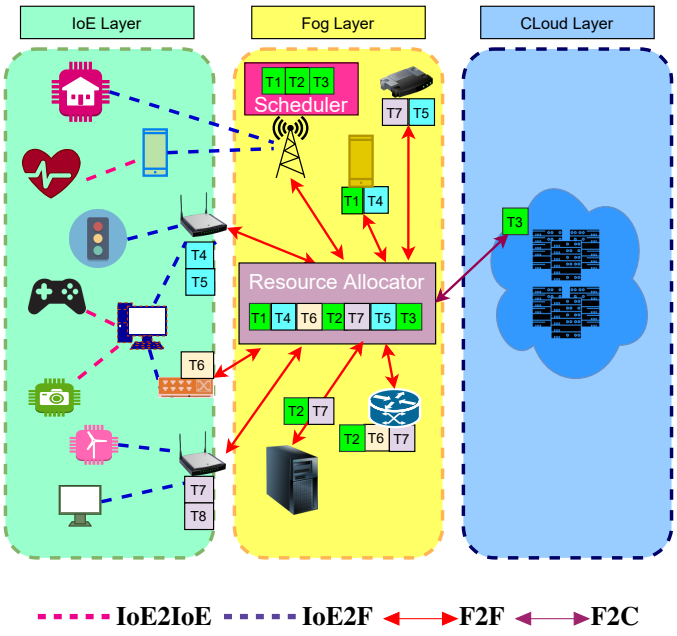



Fig. 1. Fog Computing Architecture. F = Fog Node; C = Cloud; IoE2F = IoE-to-Fog; IoE2IoE = IoE-to-IoE; F2F = Fog-to-Fog; F2C = Fog to Cloud

[10] are presented. Among these approaches, fog computing, introduced by Cisco, has gained the most attention due to its greater support for application processing, mobility, low-energy consumption, heterogeneity, geographical distribution, thus achieving significant improvement of Quality of Service (QoS) expectations [11].

### *A. Fog Computing Paradigm and Benefits*

Fog computing is a novel decentralized computing paradigm that complements cloud computing by using an additional layer of fog devices between the cloud data centres and the end-users to provide computing, storage, network, analytics, and data management in a closer vicinity of the edge at various fog nodes and IoE devices [12]. This has led to the use of a decentralized Fog-IoT Framework (FIoT) for the IoT model that supports real-time services, mobility, geographic distribution, minimizes service latency, energy consumption, cost, and network traffic. In Fig. 1, we present the multi-layer, bi-directional, and decentralized architecture of FIoT that consists of three tiers.

- **Cloud Layer:** This layer is the topmost layer that includes a remote centralized cloud responsible for performing long-term analysis and decision-making.
- **Fog Layer:** This layer is the intermediate layer that consists of three sub-layers in which the fog nodes are deployed as given below:
  1) Inner Edge: The inner edge layer consists of WAN and Metropolitan Area Network (MAN) data centers, Internet Service Providers (ISPs), etc., that help in connecting local networks to global networks.
  2) Middle Edge: The middle layer consists of Local Area Networks (LANS) and cellular network.
  3) Outer Edge: The outer edge includes different types of devices e.g. hubs, switches, Raspberry Pis, nano-servers, mobile phones, radio towers, etc. [13]. These devices communicate with upper-level fog devices via Fog to Fog (F2F) link using ZigBee or Z-wave [14]. These outer edge devices are arranged in two sub layers as mentioned below.
     - Integrated Devices: These devices have adequate memory, storage, and processing power, along with networking capabilities, for example, CPU-based smartphones and tablets.
     - IP Gateway Devices: IP Gateways are used as intermediate devices between the end and the middle edge layered devices e.g. Hubs or IP gateway devices.
- **End Devices Layer:** This layer consists of heterogeneous and resource-constrained devices, such as sensors, actuators, and controllers, that can communicate through IoE2IoE connection using short-range waves like RFID, Bluetooth, and Radio Frequency [2]. Sensors receive data from the physical environment, convert it into signals, and then send those signals to nearby fog nodes for further processing. Once processed, feedback is sent back to actuators that take actions accordingly.

In contrast to the cloud and inner edge-fog nodes, the devices in the outer edge and end device layer are resource and energy-constrained and heterogeneous (having various hardware specifications, communication protocols, and architectures) [15]. These heterogeneous resource-constrained fog nodes and highly unpredictable nature of the fog computing environment make resource management a challenging issue. The challenges includes service placement, resource discovery, service migration, load balancing, task scheduling, resource allocation, energy efficiency, and QoS [16]. Among these resource management issues, resource and task scheduling challenges are significant due to their effect on enhancing the overall system's performance.

### *B. Scheduling in Fog Computing*

The IoE applications generate a large number of variable-length tasks that often require priority based execution. But, end devices of the network are heterogeneous and resource-constrained. These tasks have to compete for constrained resources of these heterogeneous devices in a heterogeneous environment [17]. Therefore, to execute these jobs according to their resource demands, appropriate nodes having sufficient resources should be allocated. Furthermore, an efficient and convenient ordering of these tasks on heterogeneous fog nodes with available resources can enhance efficiency and accuracy of the task execution process and maximization of resource utilization like processor, memory, bandwidth, and minimizing delay, cost, and energy consumption.

In fog computing, three major scheduling issues are resource allocation, task scheduling, and workflow scheduling [18] as described below:

- **Resource Allocation:** Resource allocation aims at the optimal allocation of a set of tasks {T1, T2,..., Tn} with different QoS requirements to a set of densely distributed heterogeneous fog nodes {F1, F2,..., Fm} to yield faster

response, improved resource utilization, decreased energy consumption, makespan and cost [19].

- **Task scheduling:** The fog devices receive a vast number of heterogeneous tasks for processing from different fog applications (latency-sensitive or delay-tolerant) that are dynamic, with varying lengths, and often require priority-based execution. These tasks wait in a ready queue for execution on resource-constrained fog nodes. Therefore, an efficient, fast, and convenient way of sequencing these tasks according to their criticality to maximize resource usage and minimize delay, cost, and energy is of great importance.
- **Workflow scheduling:** The IoE devices generate jobs that can be decomposed into a series of tasks. These tasks can be either independent or dependent [20]. The independent tasks do not affect the other tasks and they are not affected by them either. Therefore, they are independent of execution order. However, the dependent tasks can execute only after the completion of their parent tasks. Dependent task scheduling, also known as workflow scheduling, is described by the Directed Acyclic Graph (DAG) or workflows. The objective of workflow scheduling is to distribute tasks onto heterogeneous fog nodes and decide an execution sequence of all task in a workflow according to their dependencies along with minimization of their makespan. Workflow scheduling in a complex-distributed environment is an NP-complete problem and is extensively studied by researchers [21].

### *C. Challenges in Resource Allocation, Task Scheduling, and Workflow Scheduling*

The challenges faced by resource allocation, task scheduling and Workflow techniques in fog computing environment are discussed below:

- **Outer-edge Fog Nodes:** In contrast to the cloud and inner-edge fog nodes, the devices in the outer edge and end-device layer have [15]:
  - **Heterogeneity:** various hardware specifications, communication protocols, and architectures
  - **Resource-constraint:** limited processing capability and storage
  - **Energy-constraint:** limited power
  - **Distributed nature:** dense geo-spatial distributions
  - **Dynamic workloads:** different workloads Optimal allocation of a set of tasks to a large number of geographically-distributed fog nodes is a challenging issue.
- **Heterogeneous Fog Applications:** Fog applications vary in nature as some applications are real-time while some others are delay-tolerant. The real-time applications have low latency requirements, therefore, they should be executed on a priority basis. Allocating a large number of heterogeneous tasks to an appropriate fog node, from a set of heterogeneous and resource-constrained fog nodes, makes resource allocation an NP-hard problem [22].
- **Stochastic Environment:** The fog-cloud environment is stochastic in many ways. For example, a task's arrival rate, duration, as well as computational requirements vary randomly and are not known in advance. Therefore, allocating the limited resources of fog nodes to these IoE-generated workloads according to their computational needs is another challenge.
- **Mobility:** Mobility is an essential part of many fog applications for improving user experience. Mobility causes several challenges like handling user's preferences, time and distance constraints [23]. Therefore, allocation and scheduling in such a dynamic environment become even more challenging to accomplish user requests according to their preferences and QoS requirements for real-time applications [24].

The resource-constrained, the heterogeneity, unpredictable arrival rate and vast number of tasks to be completed make resource and task scheduling a complex issue. To overcome this problem, many research efforts have been done to develop efficient scheduling techniques to optimize resource utilization and various performance metrics. However, there is need of a deep and up-to-date review of proposed research methods to find the recent advances in scheduling in fog environment.

### *D. Comparison with Existing Surveys*

Despite the significance of the task scheduling issue, we have found only four surveys that present the overview of the efforts done for scheduling in fog. Hosseinioun et al. [25], study the related task scheduling works in fog computing between 2015 to 2018. They classify the algorithms as static and dynamic. The static algorithms are further classified as heuristic and meta-heuristic algorithms, while real-time scheduling algorithms are discussed as dynamic algorithms. The scheduling algorithms are compared with one another based on certain QoS factors. They also stress the strengths, weaknesses, and simulation environment for each study. However, the survey does not include the recent studies and the information related to the application scheduling environment.

In another survey, Yang and Rahmani [26] review the task scheduling mechanisms in fog computing. The authors categorize the scheduling algorithms as heuristic and meta-heuristic mechanisms. For each algorithm, they present the main features and the environment. The algorithms are compared with one another based on the performance and resource utilization metrics. The simulation environment used in different related studies is not discussed. The authors partially review the related literature, and many recent studies are also not included.

Alizadeh [27] point out a comprehensive survey of different task scheduling algorithms and their strengths and weaknesses from 2015 to 2020. They classify the task scheduling algorithms as a static, dynamic, heuristic, and hybrid algorithms. They also discuss the scheduling type, application environment, and the simulator used in each study. They provide a neat comparison of the selected studies stressing different performance metrics. The selected literature lacks an appropriate categorization and review of many intelligent learning-based techniques.

Matrouk and Alatoun present a survey on scheduling algorithms in fog computing [28] in which they discuss the algorithms for resource allocation and scheduling issues. They

classify the related literature according to a focused scheduling problem like task scheduling, resource scheduling, resource allocation, job scheduling, and workflow scheduling. For every related study, the authors discuss its merits and demerits. They also compare the proposed algorithms according to specific metrics and evaluation tools. The work suffered from a lack of analysis and study on the fog computing and IoE application environment and missed comprehensive application metric comparisons, which we address in this survey. Table I summarizes the advantages and limitations of existing surveys.

The existing surveys review the mono or bi-objective optimization-oriented scheduling techniques, which are based on prior details. As the fog computing environment is dynamic; therefore, recently the researchers focus on designing dynamic, intelligent, efficient, and multi-objective task scheduling algorithms that can jointly optimize performance and resource utilization metrics along with task deadlines constraints. These surveys do not include the self-learning, intelligent, and dynamic task scheduling techniques like Reinforcement Learning (RL), Deep Reinforcement Learning (DRL), and machine learning-based techniques. These techniques may become more successful because of their ability to handle the uncertain environment, self-learning ability, computational efficiency, and adaptive nature [29].

### *E. Motivation and Goal of the paper*

As explained in the previous subsections, efficient scheduling algorithms can significantly improve the system performance by optimizing resource utilization and other performance metrics. Resource-constrained fog nodes and latency-sensitive nature of some fog applications make scheduling more significant. A variety of research efforts have been done for resource allocation, task scheduling and workflow scheduling in fog computing. However, a thorough review of up-to-date research methods proposed to date was still required to find the recent advances in scheduling in fog environment.

This paper addresses three main scheduling topics namely task scheduling, workflow scheduling, and resource allocation, as shown in figure Fig. 2 with a detailed survey of the research work done so far to optimize the resource utilization and performance metrics. As compared to the existing surveys, this paper reviews most of the techniques presented so far for scheduling and resource allocation in fog computing with a focus on learning-based dynamic algorithms. Our survey provides a detailed overview of the scheduling techniques, their focused metrics, and evaluation tools.

In this survey, we discuss and analyze the algorithms for three types of scheduling and classify these algorithms as Traditional, Heuristic, Hyper Heuristic, Hybrid Heuristic, Meta-Heuristic, Fuzzy based, Reinforcement Learning, and Deep Reinforcement Learning-based. Through analyzing the results discussed in recent research papers, we create an understanding of the subject for the reader. In addition, the research gaps and future research directions are also discussed. In the end, we present the strengths and weaknesses of different algorithms to help the researchers in selecting the best existing algorithm according to their preferences or to help them in designing new ones.

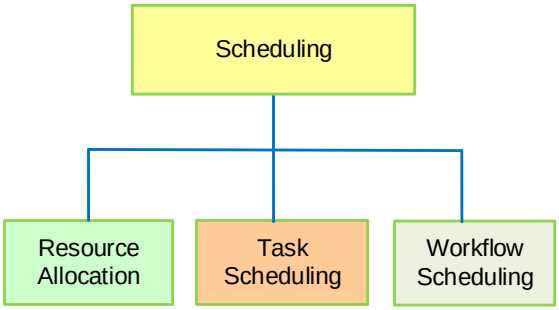



Fig. 2. Different Aspects of Scheduling.

### *F. Contributions*

The contributions of our work are six-fold as described below:

- Detailed discussion on the basics of task scheduling, workflow scheduling, resource allocation, and different optimization metrics used for evaluation of these algorithms
- Classification and comprehensive review of existing scheduling algorithms particularly focusing on intelligent dynamic scheduling techniques based on machine learning, fuzzy logic, reinforcement learning and deep reinforcement learning
- Description of the strengths and weaknesses of task scheduling, workflow scheduling and resource allocation algorithms
- Comparison of different scheduling algorithms considering different optimization metrics
- Presentation of various simulation environments and tools used in different studies
- Identification of the research gaps and challenges for task scheduling and resource allocation in fog computing for future research work in this field

### *G. Paper Organization*

The rest of the paper is organized as follows. In Section II, we present the optimization metrics for scheduling in fog environment. Section III discusses the classification of the task scheduling, resource allocation, and workflow scheduling algorithms. Section IV presents the discussion about the surveyed techniques. In Section V, we describe the issues and challenges and finally, we conclude in Section VI. Fig. 3 shows the organization of the paper.

Table II shows a list of acronyms used in this paper.

## II. Optimization Metrics

As explained above, the increasing number of latency-sensitive IoE applications, resource-constraint, and energy-limited fog devices, uncertainty and mobility support of fog computing, and meeting QoS user-requirement make optimal task scheduling a challenging issue. One of the objectives of resource allocation, task scheduling and work flow scheduling is to optimize the job execution process for maximum resource usage of fog nodes. For this purpose, in scheduling different optimization functions like mono, bi-objective, or

TABLE I
COMPARISON OF SURVEYS ON SCHEDULING.

| Reference | Objectives | Limitations |
|---|---|---|
| [25] | Review of task scheduling approaches in fog computing | Literature survey from 2015-2018 |
| [26] | Focused on task scheduling mechanisms in fog settings | Only few heuristic and meta-heuristic algorithms between 2015-2018 are discussed |
| [27] | Classification and analysis of task scheduling algorithms for fog computing | Many learning-based dynamic task allocation studies are not considered |
| [28] | Algorithms for five major scheduling issues are discussed | Recent dynamic machine learning scheduling algorithms are not included |

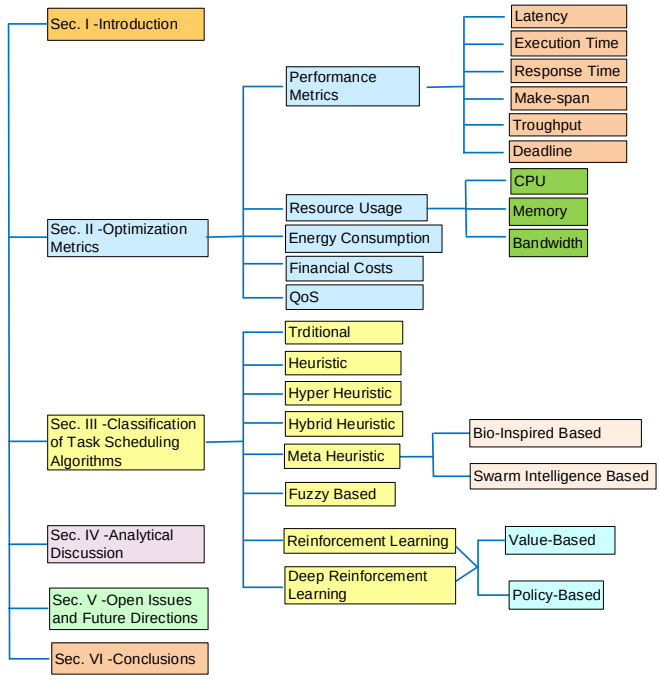



Fig. 3. Organization of the Paper.

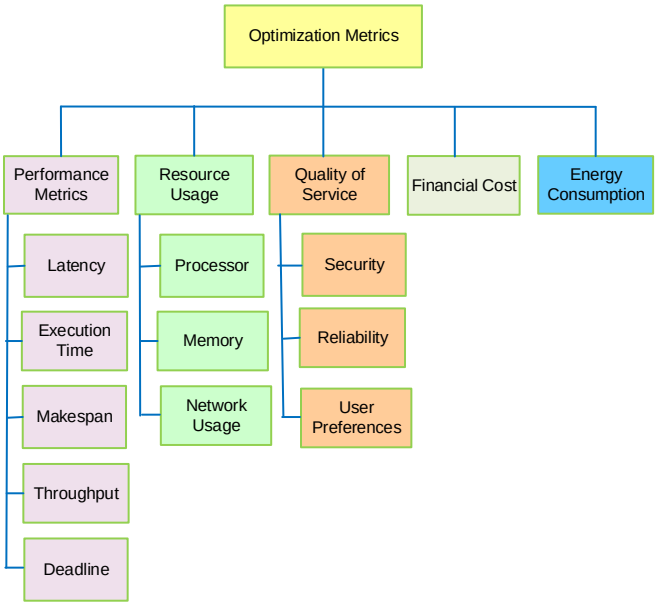



Fig. 4. Optimization Metrics.

multi-objective functions are defined to optimize resource-utilization and performance metrics like waiting time, latency, makespan, throughput, and percentage of missing deadlines of tasks. The aim of multi-objective optimization is to provide an optimal solution for more than one objectives that can be contradictory. Fig. 4 shows the optimization metrics discussed in the following subsections.

Every scheduling mechanism considers a subset of optimization metrics. In the following subsections, we present an overview of some metrics optimized in different task scheduling algorithms in cloud-fog settings.

### A. Performance Metrics

Performance metrics help in measuring the quality and efficiency of the task scheduling, workflow scheduling, and allocation process. Different task scheduling, workflow scheduling and resource allocation mechanisms optimize different performance metrics. In this subsection, we discuss the commonly used performance metrics optimized in different algorithms.

- **Latency:** Latency is one of the most important metrics to measure the performance of any task scheduling algorithm. Latency is also known as delay or response time. The overall latency is the summation of transmission latency and computational latency [30]. The transmission latency is the communication delay to transfer data between resources and the computational latency is the time taken for the task to be processed. The computational latency of any task is calculated using Equation (1) given below:

$$Latency_i = TLi + CLi \tag{1}$$

where $Latency_i$ is the latency, $TL_i$ is the transmission latency and $CL_i$ is the computational latency of any task i.

- **Execution Time:** The time taken by a system to execute the task is known as execution time. The waiting time for I/O and other executing tasks is not included in CPU or execution time [31].
The execution time is computed using Equation (2) as follows:

$$ExeTime_i = FT_i + ST_i \tag{2}$$

where $ExeTime_i$ denotes the overall execution time, $CT_i$ denotes the finish time, and $ST_i$ denotes the time when task i starts execution.

TABLE II
LIST OF ACRONYMS.

| | |
|---|---|
| A3C | Asynchronous Advantage Actor Critic |
| ACO | Ant Colony Optimization |
| AMO | Ant-Mating Optimization |
| BLA | Bee Life Algorithm |
| BPP | Bin Packing Penalty |
| CIoT | Cloud-centric Internet of Things |
| CNN | Convolutional Neural Networks |
| CPS | Cyber-Physical System |
| DAG | Directed Acyclic Graph |
| DBNs | Deep Belief Networks |
| DDQN | Double DQN |
| DNN | Deep Neural Network |
| DOTS | Delay-Optimal Task Scheduling |
| DQN | Deep Q-network |
| DRF | Dominant Resource Fairness |
| DRL | Deep Reinforcement Learning |
| DVFS | Dynamic Voltage and Frequency Scaling |
| EDA | Estimation of Distribution Algorithm |
| EDF | Earliest Deadline First |
| EFT | Earliest estimated Finish Time |
| F2F | Fog to Fog |
| FIoT | Fog-IoT |
| FBRC | Fog-based Region and Cloud |
| FCFS | First-Come-First-Served |
| FCM | Fuzzy C-Mean Clustering |
| GA | Genetic Algorithm |
| GfE | Greedy for Energy |
| GKS | Greedy Knapsack-based Scheduling |
| GP | Genetic Programming |
| GSP | Greatest Satisfactory Proportion |
| HEFT | Heterogeneous Earliest Finish Time |
| IaaS | Infrastructure as a Service |
| IACO | Improved Ant Colony optimization |
| ILP | Integer Linear Programming |
| IoE | Internet of Everything |
| IoT | Internet of Things |
| IPSO | Improved Particle Swarm Optimization |
| ISPs | Internet Service Providers |
| JSSP | Job-Shop Scheduling Problem |
| KNN | K-Nearest Neighbours |
| LSP | Least Satisfactory Proportion |
| LSTM | Long-short Term Memory |
| MAN | Metropolitan Area Network |
| MARL | Multi-Agent Reinforcement Learning |
| MCT | Minimum Completion Time algorithm |
| MFO | Month-Flame Optimization |
| MINLP | Mixed-Integer Nonlinear Programming |
| MLPs | Multilayer Perceptrons |
| MobMBAR | Mobility-Aware Modified Balance Reduced |
| PERA | Prioritized Efficient Resource Algorithm |
| PGA | Prioritized Genetic Algorithm |
| Po2C | Power of 2 Choices |
| PPO | Proximal Policy Optimization |
| PSO | Particle Swarm Optimization |
| PTPN | Priced Timed Petri Nets |
| QoE | Quality of Experience |
| QoS | Quality of Service |
| R2N2 | Residual Recurrent Neural Network |
| ReLU | Rectified Linear Unit |
| RFID | Radio Frequency Identification |
| RFN | Rank Fog Nodes |
| RL | Reinforcement Learning |
| RNN | Recurrent Neural Networks |
| RR | Round Robin |
| RSS | Received Signal Strength |
| SA | Simulated Annealing |
| SARSA | State–Action–Reward–State–Action |
| SET | Shortest Execution Time |
| SJF | Shortest Job First |
| SVM | Support Vector Machine |
| TCaS | Time-Cost aware Scheduling |
| TRPO | Trust Region Policy Optimization |
| VMs | Virtual Machines |
| VNs | Voluntary Nodes |
| WOA | Whale Optimization Algorithm |
| WSM | Weighted Sum Method |
| WSN | Wireless Sensor Network |

- **Makespan:** Makespan is a significant objective of task scheduling that indicates the overall time required to finish a complete workflow [32]. It can be computed by considering the last task's completion time and the time of submission of workflow. The minimization of makespan results in a fast execution of the applications. Makespan can be computed using Equation (3) as follows:

$$Makespan = CT_l - ST_f \tag{3}$$

where $CT_l$ denotes the time when last task is completed and $ST_f$ denotes the starting time of the first task.

- **Throughput:** Throughput of a system is the number of tasks completed per unit time [33]. Equation (4) can be used to compute throughput as follows:

$$Throughput = \frac{\text{Number of tasks}}{\text{Makespan}} \tag{4}$$

- **Deadline:** Deadline is the time duration from task submission to the time till it must be completed. In real time applications, the completion of each tasks within specified deadline is important. Especially, for the hard real-time applications like air traffic control, missing a task deadline can result in a disaster.

### *B. Resource Usage*

One significant objective of task scheduling is to increase the resource usage of fog nodes including processor, memory, storage, and network bandwidth. As the end-devices and edge-devices, like routers, gateways, Raspberry Pi etc., offer only limited computing, memory capacity, bandwidth, and battery life, so a good resource utilization strategy is necessary. Poor resource utilization may result in poor performance that can lead to application failure. A task scheduling algorithm should efficiently allocate the following resources:

- **Processor:** is the most important resource and should be carefully allocated by a task scheduler. An overloaded CPU can result in long latency of tasks that are not acceptable in the case of latency-sensitive applications like healthcare.
- **Memory:** allocation is also critical as over-booked memory can lead towards application failure.
- **Network Bandwidth:** is another significant metric. Increasing the number of IoE devices increases network traffic, resulting in congestion. This congestion badly affects the performance of the fog applications that are specifically not acceptable for latency-sensitive applications. Fog computing decreases the network usage by offloading workload on nearby fog devices that can be computed through Equation (5) as follows:

$$Network\ Usage = \sum_{k=1}^{K} L \times T\_Nw\_Sz \tag{5}$$

here $K$ is the total number of tuples, $L$ denotes the delay of each tuple, and $T\_Nw\_Sz$ denotes the network size of tuple.

### C. Energy Consumption

Energy is a necessary and scarce resource. Energy consumption is the amount of resource-energy utilized for producing output [34]. All devices like proxy, sensor, gateway, cloud, etc. consume energy even in an idle state, and this consumption increases when the device is used.

### D. Financial Costs

Financial costs include processing and communication costs like the cost of utilizing cloud or fog resources. The network usage for transferring data and energy consumption of devices also impacts the overall cost.

### E. Quality of Service

All the above attributes are quantifiable. However, some other metrics are non-quantifiable but they can affect the user experience, for example, reliability, security, user preferences, and Quality of Experience (QoE) [35]. Through optimal resource allocation, more users can be simultaneously served, resulting in a better QoS.

## III. Classification of Scheduling Methods

In this section, we present a survey of scheduling algorithms considering the different optimization metrics in a fog computing environment. The majority of these algorithms focuses on the allocation of tasks on geographically-distributed fog devices. We categorize the resource allocation and task scheduling algorithms as traditional, heuristic, hyper-heuristic, hybrid heuristic, meta-heuristic, fuzzy based, reinforcement learning, and deep reinforcement learning-based as shown in Fig. 5.

According to the taxonomy of scheduling algorithms presented in Fig. 5, the first type of algorithms are traditional algorithms. These algorithms are usually static, easy to understand, and need prior information about tasks and available resources. These algorithms are discussed in subsection III-A. Heuristic algorithms are the second type of algorithms that are commonly used for solving optimization problems. These algorithms help in finding a feasible solution in a short time. Subsection III-B presents the survey of the existing heuristic algorithms. Hyper-heuristic algorithms have gained the attention of researchers to solve problems of different domains. Subsection III-C describes the hyper-heuristic algorithms in detail. These algorithms are more general-purpose algorithms that help in selecting the best heuristic from several candidate heuristics. Subsection III-D explains the hybrid heuristic algorithms that are developed by combining different heuristic algorithms to yield a faster solution in a reasonable time. These algorithms usually perform better than single heuristic algorithms. Meta-Heuristic algorithms are found efficient for solving resource allocation in a distributed environment. These algorithms also provide a near-optimal solution in linear time. Meta-heuristic algorithms are further classified as bio-inspired or swarm intelligence-based algorithms described in subsection III-E. For handling the uncertain environment of fog computing, some researchers apply fuzzy methods that are presented in subsection III-F. They use fuzzy quantifiers and fuzzy logic for the ranking of fog nodes and optimization of performance metrics. For dynamic and uncertain fog computing environment, some researchers have successfully applied Reinforcement Learning algorithms to solve scheduling problems. Subsection III-G discusses RL-based algorithms. The RL algorithms are further classified as value-based and policy-based algorithms. Due to the dense deployment of fog nodes and the huge number of IoE requests, the search space becomes high dimensional. To design adaptive resource allocation algorithms for complex and unpredictable fog environments, some researchers apply Deep Reinforcement Learning (DRL) based algorithms. DRL algorithms can also be categorized as value-iteration and policy-iteration based algorithms explained in subsection III-H.

The following subsections describe each of the scheduling categories in more detail as shown in Fig. 5.

### A. Traditional

In this subsection, we first explain the traditional scheduling algorithms, their merits and demerits. Then, we present a review and comparison of the well-known traditional algorithms for task scheduling in fog computing.

Traditional algorithms are static algorithms in which all the information about tasks and fog resources is known in advance for a scheduling decision. These algorithms are simple and easy to understand and implement. First Come First Served (FCFS), Min-Max, Min-Min, Minimum Completion Time (MCT), and Min-Max are some well-known traditional task scheduling algorithms. FCFS and Round Robin algorithms are used for single machine while Min-Max and Min-Min algorithms are used for multiple machines.

**First-Come-First-Served (FCFS):** In FCFS, the task execution order depends upon their arrival time [36]. When a task is received, it is placed at the tail of the task queue and then the scheduler executes them according to their entry order.

**Round Robin (RR):** In this scheduling, tasks are served in FCFS order but, each task is assigned a small time interval of processor that is called time-slot or time quantum [37]. If a task completes its CPU-burst before the quantum expires, it is preempted and the processor is assigned to the succeeding task in the task queue. However, if a task does not complete its burst, it is moved at the tail of the task queue.

**Min–Min algorithm:** This algorithm selects and executes the smallest task on available machines. Therefore, it results in long delays for large tasks [38]. The algorithm then assigns a resource to the task that can give minimum completion time.

**Max–Min algorithm:** This algorithm selects and executes large tasks on available machines first so, small jobs may suffer from starvation.

**Minimum Completion Time algorithm:** Minimum Completion Time algorithm (MCT) algorithm selects and executes a task that has earliest completion time [39].

**Priority Scheduling Algorithm:** In priority scheduling, each task is assigned a priority and then executed according to its priority number [40]. All tasks with equal priority number

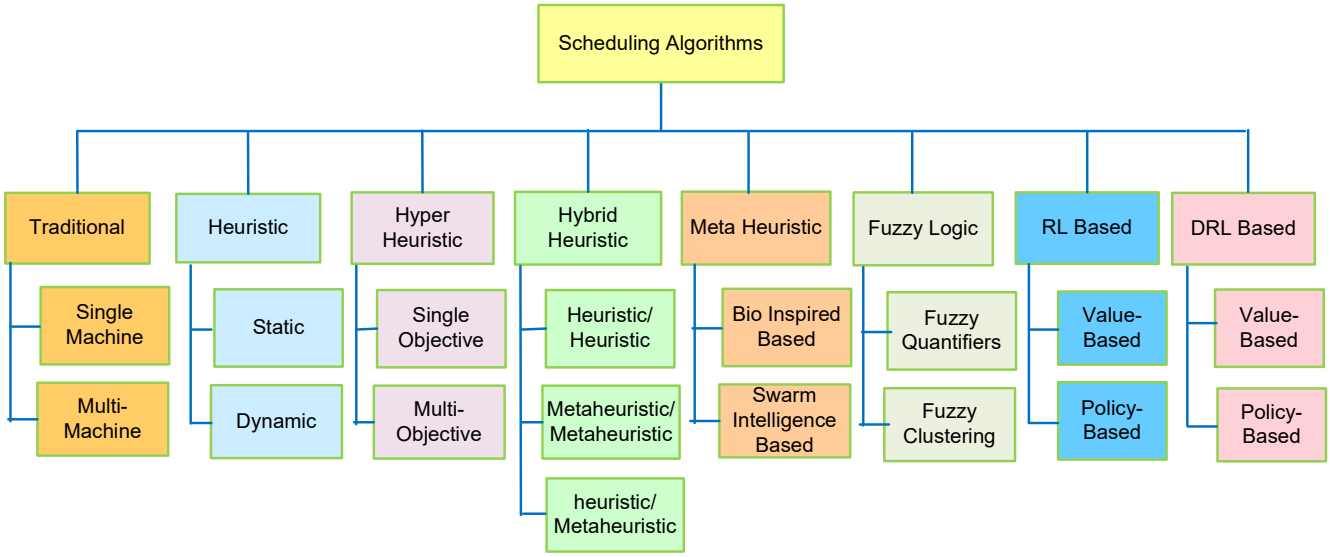



Fig. 5. A taxonomy of resource allocation, task scheduling and workflow scheduling algorithms in fog computing environment.

are executed in an FCFS manner. For example, Shortest Job First (SJF) algorithm prioritizes the tasks according to their CPU burst.

These static algorithms are simple and easy to implement and focus on optimizing latency, makespan, execution time, and resource optimization. Below is the survey of the well-known traditional algorithms of task scheduling on fog nodes.

- **Task Scheduling** Harshit Gupta et al. use FCFS scheduling algorithm for task scheduling on two different types of applications using iFogSim [41]. iFogSim is a simulator that executes at the top of CloudSim and helps in computing application latency, energy consumption, and network usage. The authors compute these evaluation metrics for both applications. In case of non-availability of processing power, a task is sent to the scheduling queue. The authors prove that the edge-word placement of modules yields better results as compared to the cloud placement in terms of network use, delays, and energy consumed. Tejaswini et al. [42] propose a Prioritized Efficient Resource Algorithm (PERA) to decrease response time and cost in fog-cloud architecture. Upon task submission, the task priority is computed according to its deadline. Once the task is prioritized, it is sent towards the fog layer for execution. The fog layer is composed of many micro data centers and fog devices that can communicate with one another. If none of the fog layer data centers is available to handle a task due to resource limitation, it is sent to the cloud. The authors implement their algorithm using the CloudAnalyst simulator. Their performance results prove that prioritized scheduling can minimize response time and cost because the tasks are prioritized according to their delay-tolerant requirements. The limitation of their work is that they do not dynamically compute task priority and ignore important metrics like energy consumption and network usage.
  In a study, Mtshali et al. present an application scheduling technique to investigate the most efficient task scheduling algorithm to minimize energy consumption, network usage, and average task execution latency for real-time applications [43]. They implement four task scheduling algorithms FCFS, SJF, Round-Robin, and Genetic Programming (GP), using the iFogSim simulator. According to their analysis, FCFS's performance is the best in minimization of latency and energy consumption than the performances of other algorithms.
  Alsmadi et al. propose a weighted round-robin (WRR) task scheduling algorithm for a smart city [44]. The algorithm selects the optimal fog node for executing a pre-emptive task based on available computing capacity and residual energy. If the task cannot be executed on a fog node, it is moved to other fog nodes or cloud. The spanning-tree protocol is used to collect and route the data in a fog network. For simulation, the authors use iFogSim and NS3 simulators. The authors compare their proposed algorithm with the conventional task scheduling algorithm considering the optimization of throughput and latency.
- **Resource Allocation:** Bittencourt et al. [45] apply three different scheduling algorithms namely: FCFS, concurrent, and delay-priority for mobility-aware scheduling in fog environment to analyze both real-time and delay-tolerant applications. They take two case studies: one of delay-tolerant application and the other of a near real-time application. They use iFogSim for simulation and compute loop delay as well as network usage for different configurations in order to show that scheduling policies should be designed for appropriate handling of the mobility and latency requirements of the application. A weakness of the research is the high computational cost.

A comparison of traditional task scheduling and resource allocation techniques considering different optimization metrics along with the simulation tool used are presented in Table III, where the tick marks indicate the focused criteria of the researchers and the crosses show the ignored metrics.

TABLE III
COMPARISON OF TRADITIONAL SCHEDULING ALGORITHMS.

| Type | Name | Latency | Makespan | Execution Time | Network Usage | Energy Consumption | Cost | QoS | Evaluation Tool | Reference |
|---|---|---|---|---|---|---|---|---|---|---|
| Traditional | FCFS | ✓ | × | × | ✓ | ✓ | × | × | iFogSim | [41] |
| Priority | PERA | ✓ | × | × | × | × | ✓ | × | CloudAnalyst | [42] |
| Traditional | WRR | ✓ | × | × | × | × | × | × | iFogSim, NS3 | [44] |
| Traditional | FCFS, Delay-Priority, concurrent | ✓ | × | × | ✓ | ✓ | × | × | iFogSim | [45] |

Table IV lists the scheduling techniques and the environment used by researchers in their studies.

As the traditional algorithms are static, these algorithms are better when workloads do not frequently vary. These algorithms are not suitable for the unpredictable fog environments with varying workloads.

### *B. Heuristics*

In this sub-section, we present the review and comparison of different heuristic algorithms used by researchers to solve resource scheduling problems along with their focused metrics. Heuristic algorithms are flexible and well-suited methods for performance optimization of the scheduling problems [46] that aim to provide an optimal solution in a short time. These algorithms help in solving NP-complete problems. The scheduling techniques used by researchers in their studies related to resource scheduling are discussed below.

- **Task Scheduling:** In [47] Jamil et al. design a heuristic-based fog node task scheduler for IoE-service provisioning. The authors present a Smart Healthcare case study to optimally schedule a variety of incoming tasks with different computational needs, like latency-sensitive, delay-tolerant etc., on fog nodes. Their algorithm aims for the optimization of delay, energy consumption, and better resource utilization. They uses iFogsim for simulating the fog environment. Their algorithm performs better in minimization latency and network usage than FCFS. However, the long tasks can starve using their approach.
- **Resource Allocation:** Lina et al. [19] present Priced Timed Petri Nets (PTPN) algorithm based on task's completion time along with the credibility of fog devices to optimize resource usage and enhance QoS requirements of a user. In their work, the user can select the appropriate resource from the allocated resource group and then using the credibility of both user and available resources task allocation is done considering the cost of time. Their calculations deviate because the credibility of both users and resources is dynamic and the same credibility of different users is assigned to multiple groups. The authors ignore average completion time, network usage, and fairness.

  Another heuristic-based resource allocation algorithm, Fog-based Region and Cloud (FBRC), is designed by Thanh and Doan [48] that aims at decreasing latency in a fog-cloud environment. They present a fog-based region architecture for the allocation of tasks to fog regions and clouds. The scheduling problem is considered as an Integer problem to reduce completion time in which tasks are allocated based on their probability distribution. The authors ignore important metrics like energy consumption and execution costs of tasks.

  To achieve energy-efficient resource allocation, Yang et al. [49] describe Maximal Energy-Efficient Task Scheduling (MEETS) algorithm that utilizes efficient cooperation between neighborhood nodes through cognitive spectrum access techniques. They formulate the optimization problem as multi-nodes fractional programming. After formulation, MEETS algorithm is applied to offload tasks on homogeneous fog networks to enhance energy efficiency. Their simulation results indicate that MEETS is more energy-efficient than other algorithms. Also, the available spectrum bandwidth and the probability of spectrum access greatly affect the offloading decision. However, the work only focuses on overall energy optimization while ignoring energy conversions across the fog and IoE layers.

  Farooq et al. [50] design two scheduling algorithms namely Min-CCV and Min-V in order to jointly minimize computation, communication and violation cast delay for fog-cloud environment. They formulate resource allocation problem as a Mixed-Integer Nonlinear Programming (MINLP). After formulation, Min-CCV algorithm is used to minimize computation, communication and violation cast while Min-V is used to minimize deadline violation cost. The experimental study is done by varying number of tasks, fog and cloud nodes. Their experimental results show better deadline satisfaction task rates and minimized total cost than those of a genetic-based algorithm. The main drawback of the algorithm is that increasing the number of devices results in increased energy consumption and thus affecting the overall system cost.

  For dynamic resource allocation to multiple real-time workflows in fog-cloud environment, Stavrinides in [51] describes a heuristic algorithm (Hybrid-EDF). In the study, the tasks are allocated on resources according to their computational demands. The communication-intensive tasks with low computation requirements are scheduled at fog nodes while the computation-intensive tasks with low communication demands are executed at cloud nodes. The scheduling strategy works in two steps. Firstly, the tasks are prioritized according to their deadlines using Earliest Deadline First (EDF) rule. Whereas, in the second step, VMs are allocated

TABLE IV
SCHEDULING TYPE FOCUSED BY TRADITIONAL ALGORITHMS.

| Reference | Environment | Task Scheduling | Resource Allocation | Workflow Scheduling |
|---|---|---|---|---|
| [41] | fog-cloud | ✓ | × | × |
| [42] | fog-cloud | ✓ | × | × |
| [44] | fog-cloud | ✓ | × | × |
| [45] | fog-cloud | × | ✓ | × |

to each task based on the Earliest estimated Finish Time (EFT). The results show that the scheduling strategy reduces the deadline-miss ratio but results in increased monetary cost.

For real-time resource allocation, Auluck et al. [52] present two heuristic algorithms generated by autonomous cars. The study is based on an embedded fog-cloud framework in which real-time tasks are categorized as hard, firm, or soft. The hard real-time tasks are the most latency-sensitive and therefore, assigned to embedded fog processors. The firm real-time tasks are less sensitive and are scheduled at fog processor while the soft ones are sent to cloud processor for scheduling. The EDF algorithm is used to schedule individual tasks on an assigned processor. In EDF, initially, all the tasks are ordered according to the increasing order of their deadlines. Then, a static LFC (Local, Fog, Cloud) algorithm is used for scheduling the tasks in queues according to their delay tolerances. The iFogSim simulator is used for simulation. Their results show the minimization of overall communication delay.

An efficient IoT architecture along with mobility-aware scheduling algorithm is proposed by Abdelmoneem et al. [53] for latency-sensitive healthcare applications. The authors use adaptive Received Signal Strength (RSS)-based handoff technique for supporting a patient's mobility. The approach works in two steps: The ranking step and the scheduling step. In the first step, the tasks are ranked according to the emergency-level using the Weighted Sum Method (WSM). In the second phase, the heuristic-based algorithm Mobility Aware Modified Balance Reduced (MobMBAR) is used to schedule tasks on computing nodes based on the movement of patients. The authors minimize the scheduling and response time of tasks. iFogSim is used for evaluating MobMBAR and the results are compared with three other algorithms. Their results indicate that their algorithm outperforms other algorithms in reducing makespan and consumed energy.

Another resource allocation mechanism (TIPS) is presented by Zeng et al. [54] using task image placement for minimization of task completion time along with better user-experience. Initially, they stored task image on the storage server. The processing is performed on embedded client and fog devices. They use Mixed-Integer Nonlinear Programming (MINLP) to formulate their problem. The presented three-step heuristic algorithm minimizes overall completion time by dividing the problem into three subproblems: processing, I/O, and transmission time. Their results show that their algorithm performs better than greedy algorithms but the authors ignore mobility of user and financial cost.

Pooranian at el. [55] propose a scheduling algorithm for allocating resources in fog computing to reduce latency and consumed energy. They consider scheduling as a Bin Packing Penalty (BPP)-aware problem in which bins are fog servers and the Virtual Machines (VMs) are packs. The servers are penalized and rewarded based on wasted energy, time, and frequency. They use "penalty and reward policy" for the optimization of energy consumption. The algorithm computes the total number of VMs that could be allocated to a server for a certain time duration. The penalty and reward are awarded to reduce consumed energy.

Zhang et al. in [56] present Delay-Optimal Task Scheduling (DOTS) algorithm based on capabilities of the Voluntary Nodes (VNs). They develop a general analytical model for the problem and propose algorithm for minimizing overall task computational latency. Simulation is done to obtain numerical results that indicate that DOTS can effectively offload tasks on VMs to reduce processing delay as compared to command-mode offloading along with minimized energy consumption and high fairness level among fog nodes.

To achieve balanced performance among delay and energy consumption, Yang and fellows [57] present a new energy-aware framework for similar fog networks. The authors use a control parameter V for characterization between delay-energy tradeoff and present the DEBTS algorithm based on Lyapunov optimization techniques for minimizing overall energy consumption and average service latency. Lyapunov optimization aims to stabilize queues while optimizing performance objective like minimizing average energy [58]. Numerical simulations are performed to measure the performance of the DEBTS algorithm. The results suggest that the algorithm's performance is better in minimizing delay and energy as compared to other algorithms. But the authors ignore metrics like cost and computation time.

- **Workflow Scheduling:** Xuan-Qui and Eui-Nam [59] propose a task allocation policy (TSCF) to study the contrast between task's execution times and the financial costs for the fog-cloud computing system. Their heuristic algorithm allocates fog nodes to dependent tasks. The algorithm works in two steps: Firstly, the tasks are ranked by traversing DAG. While in the second phase, the prioritized tasks are allocated to fog nodes. If these nodes are not available, the task is shifted to cloud. They use CloudSim for the evaluation of their algorithm. They compared the obtained results with three other algorithms. The results show that their presented

algorithm performs better than the other algorithms in reducing cost and execution time. They only computed the execution cost of their strategy while ignoring important metrics like cost and deadline constraint of workflows.

Guevara and Fonseca propose two schedulers CASSIA-INT and CASSIA-RR for task allocation on heterogeneous resources in the fog-cloud environment [60]. CASSIA-INT is formulated as integer linear programming while CASSIA-RR scheduler implements Randomized Rounding. The authors use QoS requirements of the applications to classify them and then assign a label. The scheduler uses workflows, labels, and resource availability for scheduling decisions. The objective of their scheduling strategy is to decrease the makespan and task's execution time. The algorithms are implemented using Java language and tested for two different types of applications. The algorithm's performance is compared with two traditional algorithms and the results show that the presented algorithm effectively minimizes the makespan and execution time of a task.

In table V, we provide a comparison of heuristic scheduling techniques based on different optimization metrics and evaluation tools. The tick marks show the intended criteria of the researchers while the crosses show the ignored metrics.

Table VI lists the techniques used by researchers for resource scheduling.

As seen in Table VI, some researchers use heuristics algorithms to solve the task scheduling problem in fog computing. The heuristics do not guarantee accurate solutions but give a solution close to the best one in a short time. The main drawback is that these solutions are greedy. Therefore, they are usually trapped in local minima problem. Also, the objective function in the heuristics aims to optimize only one or two metrics.

### C. Hyper Heuristic

In this subsection, we briefly describe hyper heuristic algorithms used in various studies, their merits and demerits, focused performance measures, and evaluation environment.

Hyper-heuristic solutions are automated search procedures for improving the generalization of searching techniques in order to solve hard computational search problems [61]. A hyper-heuristic strategy chooses one of the heuristics at each iteration from a pool of candidate heuristics. Therefore, the search space of hyper-heuristics is always inside the heuristic's search space. These strategies are applied to develop a system that can solve a class of problems instead of solving a particular problem domain.

- **Task Scheduling, Resource Allocation, and Workflow Scheduling:** In [62], Liu et al. use an improved classification mining approach to schedule IoT tasks. Their algorithm uses association rules. For mining association rules (TSFC), the authors present the I-Apriori algorithm. The generated rules and least completion time of the task are combined for prioritizing and scheduling task to a fog node. Their main focus is the minimization of task's execution and waiting time. As the approach is purely time-based, it ignores optimization of network bandwidth, completion time, and QoS.
- **Resource Allocation & Workflow Scheduling:** Sabihe Kabirzadeh [63] devises a hyper-heuristic algorithm (HH) for workflow scheduling. For efficient resource allocation at each phase, the most suitable heuristic from Particle Swarm Optimization Algorithm (PSO), Genetic Algorithm (GA), Ant Colony Optimization Algorithm (ACO), and Simulated Annealing (SA) is selected. The scheduling objectives are classified on the basis of the user and the service provider's perspective. The authors focus on the reduction of energy consumption, execution time, and cost. They use iFogSim simulator for simulations. The obtained results are compared with GA, SA, PSO, and ACO, and proven to be better in terms of reduction in cost and energy in comparison with the stated approaches.

Although hyper-heuristic algorithms can solve complex real-world problems because they offer greater levels of generalization and can solve single and multi-objective optimization problems, these algorithms are difficult to implement.

### D. Hybrid Heuristic

In this subsection, we present the brief introduction of hybrid heuristic algorithms along with their merits and demerits. After that a review of hybrid heuristic algorithms is provided.

A hybrid-heuristic algorithm combines two or more heuristic algorithms, takes complementary advantages of specific algorithms, and compensates for their weaknesses at each step. Hence, it yields better results than a single heuristic [64]. Hybrid algorithms may combine a mono-objective algorithm and a population-based algorithm, or two heuristic/ meta-heuristic algorithms.

- **Task Scheduling:** Juan Wang and Di Li [65] present a hybrid-heuristic algorithm for task scheduling. Their Hybrid-heuristic algorithm solves task scheduling problems using Improved Ant Colony optimization (IACO) and Improved Particle Swarm Optimization (IPSO) algorithm to decrease latency, energy consumed, and reliability.

The hyper-heuristic and hybrid-heuristic algorithms work for the allocation of fog nodes to the tasks. Most of these algorithms use a single-objective function that optimizes only one of the metrics ignoring the rest, while the hybrid-heuristic algorithms optimize two of them. Most of these algorithms are implemented using iFogSim focusing on latency and energy consumption while ignoring QoS factors. In table VII, we present the comparison of hyper-heuristic and hybrid-heuristic techniques in which the crosses indicate ignored criteria and tick marks show the intended criteria.

The focused techniques of researchers are indicated by tick marks in Table VIII.

### E. Meta-heuristic

In this subsection, we discuss meta-heuristic algorithms and their further classification as: Bio-inspired (BI)-based and Swarm intelligence (SI)-based. Then, we present the survey of the meta-heuristic algorithms.

TABLE V
COMPARISON OF HEURISTIC SCHEDULING ALGORITHMS.

| Type | Name | Latency | Makespan | Network Usage | Energy Consumption | Cost | QoS | Evaluation Tool | Reference |
|---|---|---|---|---|---|---|---|---|---|
| Heuristic | SJF | ✓ | × | × | ✓ | × | × | iFogSim | [47] |
| PTPN | RASPTPN | × | × | × | ✓ | × | ✓ | Simulation(NA) | [19] |
| Heuristic | FBRC | × | ✓ | × | × | × | × | iFogSim | [48] |
| Heuristic | MEETS | × | × | × | × | ✓ | × | Simulation(NA) | [49] |
| Heuristic | Min-CCV, Min-V | × | × | × | × | ✓ | ✓ | Matlab | [50] |
| Heuristic | Hybrid-EDF | ✓ | × | × | × | × | ✓ | Simulation(NA) | [51] |
| Heuristic | EDF & static LFC | ✓ | ✓ | × | × | ✓ | × | iFogSim | [52] |
| Heuristic | MobMBAR | ✓ | × | × | × | × | × | iFogSim | [53] |
| MINLP | TIPS | × | ✓ | × | | × | ✓ | Simulation(NA) | [54] |
| Heuristic | BPP | × | × | × | × | ✓ | × | iFogSim | [55] |
| Heuristic | DOTS | ✓ | × | × | × | ✓ | × | Simulation(NA) | [56] |
| Heuristic | DEBTS | × | × | × | × | ✓ | × | Simulation(NA) | [57] |
| Heuristic | TSCF | × | × | × | × | ✓ | × | CloudSim | [59] |
| ILP | CASSIA-INT, CASSIA-RR | × | ✓ | × | × | × | ✓ | Java | [60] |

TABLE VI
SCHEDULING TYPE FOCUSED BY HEURISTIC ALGORITHMS.

| Reference | Environment | Task Scheduling | Resource Allocation | Workflow Scheduling |
|---|---|---|---|---|
| [47] | fog-cloud | ✓ | × | × |
| [19] | fog | × | ✓ | × |
| [48] | fog-cloud | × | ✓ | × |
| [49] | fog | × | ✓ | × |
| [50] | fog-cloud | × | ✓ | × |
| [51] | fog-cloud | × | ✓ | × |
| [52] | IoE-fog-cloud | × | ✓ | × |
| [53] | fog-cloud | × | ✓ | × |
| [54] | edge-fog | ✓ | ✓ | × |
| [55] | fog | × | ✓ | × |
| [56] | fog | × | ✓ | × |
| [57] | fog | × | ✓ | × |
| [59] | fog-cloud | × | ✓ | ✓ |
| [60] | fog-cloud | × | ✓ | ✓ |

In the last few years, meta-heuristic algorithms have gained popularity and are commonly used for solving complex computational problems [66]. Researchers use meta-heuristic algorithms to find optimal or near-optimal solutions for task allocation in distributed computing environment [67]. These algorithms are used because they provide near-optimal solutions within a reasonable duration of time. Meta Heuristics algorithms can be categorized as:

- **Bio-inspired (BI)-based meta-heuristics:** These algorithms mimic the evolution process observed in nature to solve optimization problems. GA is an example of bio-inspired algorithms [68]. In GA, a chromosome is used to represent every individual. A chromosome comprises a binary coded string of genes. The algorithm starts with a random selection of the initial population. An objective function is used for checking the fitness of the chromosome. For generating a new population, mutation and crossover operations are performed on selected chromosomes. These steps repeat till sufficient or best offspring is found [33].
- **Swarm intelligence (SI)-based meta-heuristics:** Swarm intelligence is an evolutionary computation-based technique inspired by collective intelligence of swarms especially, from their biological systems and social-behavior models. PSO, Bee Life Algorithm (BLA), ACO, and Bat algorithm

TABLE VII
COMPARISON OF HYPER-HEURISTIC AND HYBRID-HEURISTIC SCHEDULING ALGORITHMS.

| Type | Name | Latency | Makespan | Execution Time | Network Usage | Energy Consumption | Cost | QoS | Evaluation Tool | Reference |
|---|---|---|---|---|---|---|---|---|---|---|
| Hyper Heuristic | TSFC | ✓ | × | ✓ | × | × | × | × | SimGrid | [62] |
| Hyper Heuristic | HH | × | × | ✓ | × | ✓ | × | × | iFogSim | [63] |
| Hybrid Heuristic | HH | ✓ | × | × | × | ✓ | × | × | Matlab | [65] |

TABLE VIII
SCHEDULING TYPE FOCUSED BY HYPER AND HYBRID-HEURISTIC ALGORITHMS.

| Reference | Environment | Task Scheduling | Resource Allocation | Workflow Scheduling |
|---|---|---|---|---|
| [62] | fog | ✓ | ✓ | ✓ |
| [63] | fog | × | ✓ | ✓ |
| [65] | fog-cloud | ✓ | ✓ | × |

are some examples of SI-based algorithms [69].

- **Task Scheduling:** An adaptive double fitness Genetic Task Scheduling algorithm (ADGTS) is presented by Qianyu Liu [70] for IoT task scheduling in smart cities. The main objective of the algorithm is to reduce makespan and communication cost. The algorithm allocates fog nodes to tasks by considering their computing power, communication cost, and latency requirements. The chromosome encoding scheme represents the task allocation on fog nodes and roulette selection selects the new generation. They use a single-point crossover and mutation to generate a new generation. The results indicate that ADGTS algorithm efficiently reduces cost and makespan as compared to other algorithms.
  Wang et al. [71] present a task scheduling algorithm for a decentralized fog computing environment on the basis of the immune system of the human body having self-organizing, cooperative, and robustness characteristics. The fog network is considered a collection of geographically-distributed computing devices having independent schedulers that can cooperate for synchronization to produce an optimal scheduling strategy. They implement local and meta-schedulers. The local scheduler aims to decrease the execution time and the meta-scheduler in computing nodes collects the tasks from neighborhood nodes to get the information of their computing demands like memory, processor, and bandwidth and shares this data to computing nodes. This framework helps to avoid conflicting scheduling decisions and single point of failure of schedulers in a decentralizing environment. The comparison of their algorithm is done with five other algorithms. Their results show that the algorithm results in efficiently minimizing task finishing time as compared to the other algorithms.
  Another algorithm to solve the task allocation issue for Bag-of-Tasks applications in a fog-cloud environment is described by Binh Minh Nguyen at al. [72]. The main objective of an algorithm named Time-Cost aware Scheduling (TCaS) is to minimize the execution time of tasks and operating costs. Every chromosome shows the assignment of a task to a fog node. Mutation and two-point crossover are used for the generation of a new population. They use iFogSim for simulations. Their algorithm's performance is compared with two other algorithms. Their obtained results indicate that the algorithm attains a better trade-off between cost and makespan. The authors, however, ignore some important metrics like network usage and energy consumption.
  Salim Bitam [73] presents a static task scheduling algorithm using BLA for allocation of tasks on fog devices. In this algorithm, the tasks are distributed on fog nodes for optimal utilization of fog computing resources along with the service-level agreements that can be fulfilled. The authors consider the scheduling problem as an Integer Linear Programming (ILP) problem to achieve a trade-off between execution time and memory requirements of computing services. The algorithm is compared with PSO algorithm. Their results indicate that the presented algorithm is more effective than PSO for reducing execution time and memory usage. The main merit of the algorithm is that it is static. However, it ignores metrics like cost and QoS.
  In 2019, Dadmehr Rahbari et al. implement Greedy Knapsack-based Scheduling (GKS) algorithm and symbiotic organism search algorithm named KnapSOS for task scheduling using iFogSim simulator [74]. The objective of the algorithm is minimization of energy consumption, latency, and network usage. The CPU usage and virtual machine bandwidth are used to create Knapsack items. They apply the algorithm on two different applications. For both these applications, they compute execution cost and energy consumption. Then they compared the obtained results with three other scheduling algorithms. Their algorithm takes a longer simulation time.
  Jayasena and Thisarasinghe use Whale Optimization Algorithm (WOA) algorithm to schedule tasks in fog computing environment [75]. WOA is a bio-inspired algorithm that uses the idea of bubble-net attacking method of the humpback whales through which they determine the prey's location and surround it [76]. Their objective function is to reduce consumed energy and execution cost. The authors use iFogSim for implementing the WOA algorithm. To analyze

the presented algorithm, they compare its performance with PSO, RR, and SJF algorithms for three different case studies. The presented algorithm is proven to perform better in terms of consumed energy and execution cost.

In [77], XU et al. present the task scheduling problem as an optimization problem for fog-cloud environment. The task scheduling technique (LBP-ACS) uses a laxity and ant-colony system to minimize energy consumption along with satisfying the task deadlines. To meet the deadline constraint of delay-sensitive tasks, the authors apply a laxity-based algorithm for task prioritization and scheduling. After that, an ant-colony system algorithm is applied to get an optimal allocation plan. CloudSim is used to simulate a fog-cloud environment as well as to evaluate the algorithm. The algorithm performance is found better as compared to three other algorithms including Heterogeneous Earliest Finish Time First (HEFT). The HEFT [78] algorithm is a popular list scheduling algorithm that prioritizes and schedules the tasks according to the finish time of tasks and GfE is mainly concerned with minimization of energy consumption.

- **Resource Allocation:** In another study, Yan Sun, Fuhong, and Haito [79] propose a resource allocation scheme. The resources are allocated to different fog clusters. It is done among fog devices that reside in a cluster. For solving multi-objective optimization problems, the authors use the idea of improved NSGA II for allocating resources among heterogeneous fog devices in a cluster. They implement their algorithm in Matlab. The results are compared with the results of two other models that indicate that the algorithm achieves smaller delays and more solidity of the execution of tasks as compared to the other techniques. The major drawback of their proposed scheme is that it lacks an appropriate technique for scheduling between fog clusters. Also, some important metrics like cost and energy are ignored in their approach.

  Considering the dynamic nature of the fog computing environment, Aburukba et al. in [80], present a model for scheduling IoE service requests from end devices. In order to minimize service latency, they present the task allocation problem as an integer linear programming problem. They implement a customized genetic algorithm as a heuristic approach to get optimal scheduling solution in a reasonable computational time. They initialize the problem size with the received IoE requests and available resources. Then, the population is generated through several candidate solutions. The fitness of every chromosome is defined using the inverse of overall delay. Crossover and mutation are used to select individuals producing new population. The performance is analyzed and compared with three other scheduling techniques. The algorithm is shown to be effective in minimizing latency and cost.

  Ghobaei-Arani et al. [81] propose a resource allocation algorithm for Cyber-Physical System (CPS) applications in fog environment using Month-Flame Optimization (MFO) algorithm. MFO algorithm is a bio-inspired population-based optimization algorithm where the main inspiration is the navigation of a month's behavior in the night around the flames [82]. The objective function of the algorithm is to decrease the execution and transfer time of tasks. Simulation is performed using iFogSim and the simulation results are compared with NSGA II, PSO, and BLA. The comparison proves that the algorithm is good for minimizing the task execution time along with better QoS for CPS applications. To fulfill the increasing needs IoE applications, WANG et al. [83] propose a resource allocation algorithm I-FASC using (I-FA) improved genetic algorithm firework algorithm. I-FA is presented by introducing the explosion radius detection mechanism of fire-works. The algorithm is compared with three other algorithms. The obtained results suggest that the presented algorithm efficiently minimizes the computing time in comparison to other algorithms.

  As in the real world, real-time applications are latency-sensitive, while some others are delay-tolerant. These different requirements increase the complexity of the fog computing environment. In order to find an optimal solution according to the needs of IoT application, Meng et al. propose an adaptive neighborhood multi-objective optimization algorithm FOG-AMOSM for fog computing [84]. The conflicting optimization objectives of the algorithm are to decrease the execution time of tasks along with minimized cost for service providers. The authors present a multi-objective evolutionary heuristic algorithm based on the method of adaptive neighborhood technique. This technique is used for better distribution of tasks on fog resources. CloudSim 5.0 is used for simulation. Their performance analysis shows that the algorithm effectively minimizes their chosen metrics in comparison with the classical Round Robin and simple GA.

  Hoseiny et al. present a priority-aware task allocation algorithm named Prioritized Genetic Algorithm (PGA) to jointly optimize computational time, energy consumption, and percentage of tasks completed before their deadlines [85]. The prioritization of tasks is based on their deadlines. Their fog broker is constituent of three components namely task receiver, task scheduler, and resource monitor. The task receiver guesses the deadlines of tasks and resource monitor evaluates the available resources. The task scheduler firstly prioritizes the tasks according to their deadlines and then genetic algorithm is applied to select appropriate fog node for every task. Simulation is done using Matlab. Their results reveal that the performance of the algorithm is better than that of Power of 2 Choices (Po2C) and Ant-Mating Optimization (AMO) algorithms.

  Potu et al. present an Extended Particle Swarm Optimization (EPSO) algorithm for resource allocation in fog-cloud environment [86]. The objective of the algorithm is to minimize task completion time and cost using the proximal gradient method. iFogSim is used for simulation purposes, and the results are compared with PSO and its variant. According to their results, EPSO works well for reducing makespan and cost.

- **Workflow Scheduling:** A cost-effective workflow scheduling algorithm is proposed by Rongbin Xu [87] based on IPSO in a fog-cloud environment. The experiment is done in Matlab, with six cloud servers and four fog servers. The authors claim to reduce the task's completion time as

compared to PSO.

A side-by-side comparison of surveyed meta-heuristic-based algorithms is presented in table IX where tick marks show the focus criteria and the crosses indicate the ignored metrics. The focused techniques of researchers are indicated by tick marks in Table X.

The surveyed meta-heuristic algorithms perform resource allocation to a set of tasks for mono or bi-objective optimization of given metrics. Although, such algorithms give an optimal solution, they require a lot of computational time. Also, these algorithms ignore QoS criteria.

### *F. Fuzzy Logic Based Algorithms*

This subsection reviews the fuzzy logic based task scheduling and resource allocation algorithms and the focused metrics.

Fuzzy logic is a powerful tool to deal with uncertainty, vague, and non-numeric information in systems and is robust to changing environment [88]. Fuzzy logic was proposed in 1965 by Lotfi Zadeh [89]. Fuzzy set theory and fuzzy quantifiers are the basic building blocks of fuzzy logic. In fuzzy sets, each element has a degree of membership, while a fuzzy quantifier is a fuzzy relationship between fuzzy sets. Fuzzy logic can effectively be used to efficiently solve dynamic real-time scheduling problems [90].

- **Resource Allocation:** Most of the research work done for resource allocation focuses on delay, energy consumption, and network usage, but unlike cloud, fog nodes are resource-constrained. Therefore, Mohammed Anis et al. use fuzzy-quantified ranking method to Rank Fog Nodes (RFN) for task allocation [91]. The ranking is done based on features of fog nodes along with user preferences using linguistic quantifiers and fuzzy-quantified propositions. Least Satisfactory Proportion (LSP) and Greatest Satisfactory Proportion (GSP) parameters are defined to differentiate among similar fog devices. The authors focus on achieving user-satisfaction along with minimization of execution delay and consumed energy.
  In [92], FCAP, a resource scheduling algorithm, is described that combines Fuzzy C-Mean Clustering (FCM) and PSO. The authors use FCM to search clusters of fog nodes and PSO for global optimization that reduces the resource search space. The algorithm works in two steps. In the first step, there is initialization of the particle population and random generation of the cluster centers. In the second step, a weighted matching method is applied to compute the fitness function. As the FCM algorithm can trap in a local optima, they combine PSO with it for global optimization and faster convergence. For the implementation of FCAP, the authors use Matlab. Their obtained results indicate that the algorithm obtains better clustering accuracy and fast convergence than that of FCM. Also, the algorithm results in better user-satisfaction as compared to the Min-min algorithm.
  For the dynamic environment of fog computing in [93], Javanmardi presents a fog resource allocator, FPFTS that jointly use meta-heuristic algorithm and uses fuzzy logic. In the designed task allocation algorithm, fuzzy-logic is used as a fitness function in the PSO algorithm. The objective of the algorithm is the optimal resource utilization of fog nodes along with the minimization of network usage and application loop delay. The algorithm is implemented using iFogsim and evaluated for different fog node configurations.
  Wu et al. propose a multi-objective Estimation of Distribution Algorithm (EDA) resource allocation algorithm for workflows based on fuzzy logic for multi-objective optimization [94]. They consider a system in which the heterogeneous processors are located at three tiers. They use fuzzy numbers for modeling uncertain IoT environments and workload characteristics. After that, they use fuzzy clustering to classify the tasks in DAG according to their characteristics. Then, these grouped tasks are allocated to relative tiers using the cluster-tier allocation rule, that is learned by fuzzy EDA. For simulations, the algorithm is developed in C++ and results are compared with HEFT and two other algorithms. According to obtained results, EDA outperforms the heuristic algorithms.
  Ali et al. present a resource allocation algorithm to schedule real-time tasks in a fog-cloud computing environment using fuzzy logic [95]. The fuzzy-based algorithm schedules the tasks on cloud and fog nodes based on task attributes (mips, storage, bandwidth), time constraints of tasks(deadline), and resource availability in the fog layer. The aim of the presented algorithm is to reduce the makespan and average turnaround time of tasks. The algorithm is implemented in the iFogSim simulator, and the comparison is done with SJF and FIFO.

Table XI lists the fuzzy-based resource allocation techniques where the tick marks show the intended criteria and the crosses indicate the metrics ignored by the researchers.

The focused techniques of researchers is indicated by check marks in Table XII.

### *G. Reinforcement Learning*

In this subsection, we explain reinforcement learning, classification of RL-based algorithms, and an overview of different RL-based algorithms of task scheduling in fog computing.

RL is a type of machine learning that has successfully been applied for solving the Job-Shop Scheduling Problem (JSSP) [96]. Reinforcement Learning is a sequential decision-making process where agent is the decision-maker that is trained to optimize its behaviour by learning from its experience while interacting with the environment. These algorithms are successful because of their ability to handle the uncertain environment, self-learning ability, computational efficiency, and adaptive nature, hence, and are well-suited to schedule a diverse-natured tasks in a fog computing environment [29].

In RL, self-learning agents can take appropriate action in a particular situation to perform a task and try to maximize the received rewards [97]. By using reinforcement learning, a system can map a situation to actions. Agents, environment, states, actions, and rewards are the five basic concepts of reinforcement learning.

1) **Agent:** Agent is a decision-maker that takes action in a particular situation. The agents choose best actions either

TABLE IX
COMPARISON OF METAHEURISTIC SCHEDULING ALGORITHMS.

| Type | Name | Latency | Makespan | Execution Time | Network Usage | Energy Consumption | Cost | QoS | Evaluation Tool | Reference |
|---|---|---|---|---|---|---|---|---|---|---|
| Genetic | ADGTS | × | ✓ | × | × | × | ✓ | × | Simulation(NA) | [70] |
| Bio Inspired | - | × | × | × | ✓ | × | × | × | Experimental | [71] |
| Genetic | TCaS | × | × | × | ✓ | × | × | ✓ | Experimental | [72] |
| Bio Inspired | BLA | × | × | ✓ | × | × | × | × | C++ | [73] |
| Knapsack | GKS | × | × | × | × | ✓ | ✓ | × | iFogSim | [74] |
| Bio Inspired | WOA | × | × | × | × | ✓ | ✓ | × | iFogSim | [75] |
| Bio inspired | LBP-ACS | ✓ | × | × | × | × | ✓ | × | CloudSim | [77] |
| Genetic | NSGA-II | ✓ | × | ✓ | × | × | × | × | Matlab | [79] |
| Genetic | - | ✓ | × | × | × | × | ✓ | × | Simulation(NA) | [80] |
| Bio Inspired | MFO | × | × | ✓ | × | × | × | ✓ | iFogSim | [81] |
| Genetic | I-FASC | ✓ | × | × | × | × | × | × | Experimental | [83] |
| Genetic | FOG-AMOSM | × | × | ✓ | × | × | ✓ | × | CloudSim | [84] |
| Genetic | PGA | ✓ | × | × | ✓ | ✓ | × | × | Matlab | [85] |
| PSO | EPSO | × | ✓ | × | × | × | ✓ | × | iFogSim | [86] |
| Bio Inspired | IPSO | × | ✓ | × | × | × | × | × | Matlab | [87] |

TABLE X
SCHEDULING TYPE FOCUSED BY META-HEURISTIC ALGORITHMS.

| Reference | Environment | Task Scheduling | Resource Allocation | Workflow Scheduling |
|---|---|---|---|---|
| [70] | fog | ✓ | × | × |
| [71] | fog | ✓ | × | × |
| [72] | fog-cloud | ✓ | × | × |
| [73] | fog | ✓ | × | × |
| [74] | fog | ✓ | × | × |
| [75] | fog | ✓ | × | × |
| [77] | fog-cloud | ✓ | × | × |
| [79] | fog | × | ✓ | × |
| [80] | fog-cloud | × | ✓ | × |
| [81] | fog | × | ✓ | × |
| [83] | fog-cloud | × | ✓ | × |
| [84] | fog | × | ✓ | × |
| [85] | fog-cloud | × | ✓ | × |
| [86] | fog-cloud | × | ✓ | × |
| [87] | fog-cloud | × | ✓ | ✓ |

TABLE XI
COMPARISON OF FUZZY-BASED SCHEDULING ALGORITHMS.

| Type | Name | Latency | Makespan | Execution Time | Network Usage | Energy Consumption | Cost | QoS | Evaluation Tool | Reference |
|---|---|---|---|---|---|---|---|---|---|---|
| Fuzzy Logic | RFN | ✓ | × | × | × | ✓ | × | × | Matlab | [91] |
| Fuzzy Clustering | FCAP | × | × | × | × | × | × | ✓ | Matlab | [92] |
| Fuzzy Logic & PSO | FPFTS | ✓ | × | × | ✓ | ✓ | × | × | iFogSim | [93] |
| Fuzzy Logic | EDA | ✓ | × | × | ✓ | ✓ | × | ✓ | iFogSim | [94] |
| Fuzzy Logic | - | ✓ | ✓ | × | × | × | × | × | iFogSim | [95] |

TABLE XII
SCHEDULING TYPE FOCUSED BY FUZZY-BASED ALGORITHMS.

| Reference | Environment | Task Scheduling | Resource Allocation | Workflow Scheduling |
|---|---|---|---|---|
| [91] | fog-cloud | × | ✓ | × |
| [92] | fog | × | ✓ | × |
| [93] | fog | × | ✓ | × |
| [94] | Thing-fog-cloud | × | ✓ | × |
| [95] | fog-cloud | × | ✓ | × |

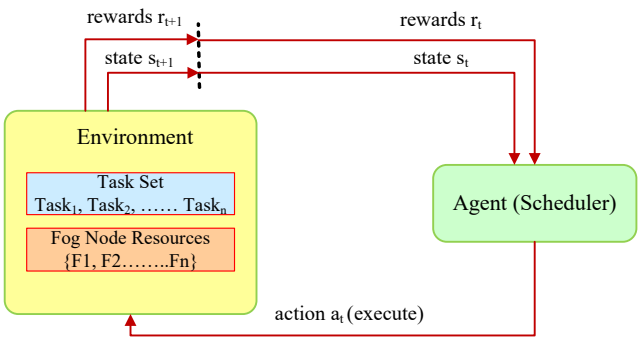



Fig. 6. Reinforcement Learning Architecture.

on the basis of their experience (exploitation) or by choosing entirely new actions (exploration). This trade-off aids the agent to learn appropriate action to take.

2) **Actions:** Actions are the set of all possible moves.
3) **Environment:** The agent observes the environment. The input of the environment is the agent's current state and action, while its output is a numeric reward and generated next state.
4) **State:** The state is the immediate situation in which the agent finds itself.
5) **Reward:** The reward is the response that describes how the agent behaved by the action taken. [98].

Fig. 6 shows the agent and environment interaction in Reinforcement learning architecture.

RL algorithms can be broadly classified as follows:

1) **Model-Based:** Model-based algorithms aim to learn the working of the environment using observations and plan a situation using model [99]. These algorithms may or may not have a policy or value function. Instead, a transition and reward function is used to search optimal policy, for example, dynamic programming. Model-based RL can further be classified as Learn Model and Model Given.
2) **Model-Free:** Model-Free algorithms do not learn from the dynamics of the environment. These algorithms are simpler and less expensive than their counterparts and can learn the optimal policy based on trial-and-error.

The three main model-free methods to solve RL problems are value-based methods, policy-based methods, and actor-critic methods [100]. These techniques are explained in the following subsections.

*1) Value-Based Methods:* Value-based methods aim to optimize the value function $V(s)$ using Bellman equation [97] in a given state. The action-value function estimates the maximum expected future return the agent will get at each state. Q-Learning is an example of off-policy value-based methods [101]. An off-policy algorithm learns the value function being independent of the policy used during training. An on-policy algorithm selects a policy depending upon the one used in gathering data. State–Action–Reward–State–Action (SARSA) is an example of on-policy value-based methods. The details of the value-based algorithms used for scheduling are given below.

Q Learning algorithm is a powerful value-based algorithm to select optimal action-selection policy using a Q function. The simple version of Q-learning sustains a lookup table of values $Q(s, a)$ where s indicates the state and a indicates the set of actions. Each entry in the table is a state-action pair used to compute the maximum future reward against an action at each state. Q-Learning algorithm uses the Bellman equation to update the value function. In Q-Learning, $Q^*(s, a)$ is the cumulative discounted reward of action a taken in any state s.

Orhean et al. use reinforcement learning for allocating tasks in a heterogeneous distributed environment [102]. They assume different machine performance and cluster status. They propose Machine Learning Box (MLBox) for the implementation of the agent using the BURLAP library. Their main objective is to design a scheduler that can optimally schedule tasks on a given cluster of machines with each machine with an internal scheduler to reduce task's execution time. In the described platform, they apply both SARSA and Q-Learning algorithms for task scheduling. The main drawback is that their approach is not suitable for a complex environment.

In 2019, Liu et al. [103] consider resource allocation problem for the IoT networks. Every edge device is considered as an agent that decides which task should be allocated on edge devices. They propose resource allocation using $\epsilon$-greedy Q-learning algorithm. Their objective is minimization of the cost of energy consumed as well as delay in task execution.

Q-Learning algorithm is simple and easy to implement, and it also yields better results [102]. But this algorithm is difficult to generalize and the agent may not select the best action in case of experiencing an unseen event. Therefore, this algorithm is not well-suited for a complex dynamic environment.

*2) Policy-Based Methods:* A policy maps agent's observed states of the environment to actions [97]. In Policy-based methods, the objective is to learn optimal policy $\pi$ with the

highest expected future rewards without maintaining the value function. A policy can be:

1) **Deterministic:** A deterministic policy maps state to actions. For each given state, the function returns a defined action to take. Deterministic policies are used in deterministic environments when there is no uncertainty. Because for a given history and action, there is a single potential observation and reward.
2) **Stochastic:** The stochastic policy is useful when the environment is uncertain. Therefore, the agent selects actions using a policy that is explained as a probability distribution over actions $\pi : \pi(s; a) \longrightarrow [0; 1]; \pi(s; a)$ the probability of performing action a under state s. For a given history and action, there are many potential observations and rewards.

Policy-based methods usually use a function approximator with some adjustable parameters, $\theta$; $\pi(s, a\theta)$.

Policy gradient algorithms are a popular type of reinforcement learning algorithms. These algorithms optimize policy parameters using gradient descent on an objective function $J$. Their goal is to estimate the gradient by analyzing the trajectories generated by following the current policy.

Qing Wu [104] propose a DAG scheduling algorithm using policy-gradient REINFORCE algorithm to execute security-sensitive tasks on trusted entities in a heterogeneous computing environment. The algorithm aims at minimization of the makespan of the tasks. They compare the obtained results of their scheduler with HEFT and CPOP for randomly-generated DAGs. The limitation of their work is that they only focus on the static problem with predefined task priorities.

### *H. Deep Learning, Deep Reinforcement Learning*

In this subsection, we introduce deep learning and deep reinforcement learning, the categories of DRL-based algorithms and a review of different DRL algorithms used by researchers.

Deep Learning is a new area of machine learning research in which feature learning and pattern classification are done through multiple processing layers [105]. In past few years, deep learning [106] has successfully been applied for image processing, sequence prediction, natural language processing, and data analysis [107]. Due to the promising results of Deep Learning, it has recently been used in various other fields like healthcare and agriculture. Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Multilayer Perceptrons (MLPs), Long-short Term Memory (LSTM), and Deep Belief Networks (DBNs) are some examples of widely-used neural networks [107]. CNN has successfully been applied for image processing as it deals with spatial distribution, and RNN has been used for natural language processing because of its ability to learn long-term dependencies.

- **Resource Allocation:** Amudha and Murali present an intelligent and novel scheduling algorithm, Modified Wake-on Reconfigurable Networks, to optimize performance using distance energy adaptive rule sets (WORN-DEAR) for patient's health monitoring [108]. They integrate deep learning algorithms in fog gateways for energy-efficient routing and scheduling without compromising latency and throughput. In their experimental setup, they implement reconfigurable software in fog gateways, and the communication between these gateways is via short-range waves. The data collected from various body sensors are moved towards the nearest fog gateways that take necessary decisions about where to process the received data. The energy-efficient path prediction for both normal and emergency data on the proposed network uses Long-Short Term Memory (LSTM). Cooja-Contiki network simulator is used for performance evaluation [109]. The work is also implemented on different testbeds, and a comparison is made with logistic regression, naïve bias, Support Vector Machine (SVM), and K-Nearest Neighbours (KNN). The demerit of the proposed work is that it ignores mobility and security factor.

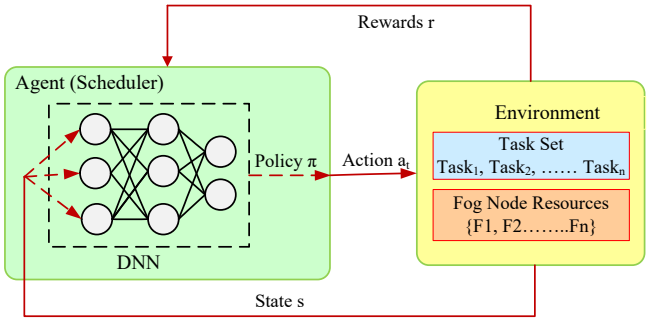



Fig. 7. Deep Reinforcement Learning Architecture.

Reinforcement learning refers to goal-oriented algorithms so, the agent aims to maximize numerical reward by taking appropriate actions. Simple RL algorithms are applied to solve few low-dimensional problems because they lack scalability. Also, RL algorithms suffer from the same complexity issues like computational and memory complexities as suffered by the other optimization algorithms [110]. The exponential explosion of states and actions is termed as "Curse of Dimensionality". Deep learning makes RL able to solve such intractable optimization problems. The fusion of deep learning and reinforcement learning named DRL has emerged as an active area of research. DRL has recently proved its capability to solve decision-making problems where mapping magnitude is too large. In the DRL model, Deep Neural Network (DNN) helps in function approximation as shown in Fig. 7.

If the state space is simple, a table can be used to store mapping of states to actions, but the solution is not feasible if the mapping magnitude is too large. Therefore, in such cases, function approximators are usually used. A function approximator contains some adjustable parameters. If a DNN is used as a function approximator, a DRL model can be created as shown in Fig. 7.

DRL algorithms are also classified as Value-Based and Policy-Based methods that are described in the following subsections.

*1) Value-Based DRL Methods:* As explained in the previous section, the value-based algorithms learn a value function that is used for defining a policy. Deep Q-network (DOQN) and Double DQN are examples of Value-Function-Based DRL algorithms.

DQN is a value-function-based DRL algorithm in which a DNN helps estimating the Q-value function [110]. Mnih et al. introduce this algorithm. It outperforms in an online setting across a variety of ATARI video games [111]. In DQN, state features are given as input to a deep neural network that outputs the Q-value function in a high-dimensional and continuous state space.

Q-Learning uses a single estimator for selection and evaluation of action so, it may select an overestimated value in case of noise. To overcome the problem of overestimation, DQN uses upward bias while Double DQN (DDQN) [112] uses two separate Q-value estimators for each variable.

- **Resource Scheduling:** Task scheduling for fog-based IoT applications using DRL is proposed by Gazori et al. [113] to decrease service latency, energy consumption, and computational cost with resource and task's deadline-constraints. They apply a DDQ-Learning based scheduling algorithm. They use gateways as schedulers that work as agents. Nodal collaboration is master-slave in which Gateways are master and fog nodes are slaves. Computing node resources are distributed among many VMs(CPU, Memory, Storage) that are allocated to incoming tasks. Their algorithm directly learns the scheduling policy from experience without having a prior knowledge of the environment. They claim to minimize energy consumption along with load-balancing on available resources.

The above review shows that DQN and Double DQN algorithms perform better as compared to the simple Q-Learning algorithm in a large and dynamic environment. But, in case of high dimensional state-action space, the algorithm converges slowly.

*2) DRL-Based Policy Gradient Methods:* Policy-Based methods aim to optimize a performance objective by finding an optimal policy, as explained in earlier sections. Policy Gradient Algorithms and Actor-Critic Algorithms are [114] commonly used as policy-based methods. Policy gradient algorithms are also classified as Deterministic and Non-Deterministic Policy Gradient (DPG) Algorithms. Gradient Ascent and Deep Deterministic Policy Gradient (DDPG) [115] are examples of Policy Gradient Methods.

In 2016, Mao et al. [116], are the first ones who claim that a system itself can learn to manage resources. They assume dynamic arrival of jobs that remain non-preempted when allocated on a cluster of resources (CPU, memory, I/O). Their main objective of the resource scheduler is to decrease average slowdown and completion time of a task. They present policy as a neural network trained in an episodic setting. In each episode, the tasks are scheduled using the policy. The authors assume that jobs arrive in the job queue after discrete time intervals. In case the job queue is full, the tasks are placed in a backlog. They used convolutions neural networks as a function approximator for state representation. They use the REINFORCE algorithm for training. However, they ignore jobs with dependent tasks.

In [117], Chen et al. also use a policy-gradient algorithm for resource allocation in multiple server clusters to reduce average job slowdown time. Their study prove that DRL outperforms conventional resource allocation algorithms in multi-resource and multi-machine environments. The authors consider $m$ machines with $d$ resources and a job queue in which jobs arrive after a discrete time step. In case the job queue is full, a job is saved in the backlog to allocate the processor in the future. They use convolutions neural networks for state representation.

- **Task Scheduling:** In [118], Ye et al. propose online task scheduling algorithm DeepRM2 and offline scheduling algorithm DeepRM-off. Their main aim is to decrease the average slowdown and completion time of the job. In the DeepRM2, job arrival is according to Poisson distribution, while in the DeepRM-off, jobs simultaneously reach a queue for a slot. They assume that there is no job preemption once it is selected. To speed up the learning process, the authors use imitation learning. For training purposes, they use the SJF scheduling algorithm. The trained neural network is then used as the initial policy for DRL. Then, the policy gradient algorithm is used in which CNN works as a function approximator.
- **Resource Allocation:** Bian et al. in [119] purpose Dominant Resource Fairness (DRF) and DRL-based solution for multi-resource fairness for resource allocation. DRF is a newly presented multi-resource fairness policy, which is a generalized form of max-min fairness policy for various resources [120] in which allocation of a user is according to the user's dominant share. Its main objective is maximization of the smallest dominant share across tasks. Their adaptive scheduling algorithm FairTS minimizes average task slowdown and schedules resources fairly among these tasks. In FairTS, scheduling is performed in two steps; In the first step, the tasks are generated as a Poisson process, while in the second step, the available resources are allocated to arrived tasks. After the allocation of the resources, no preemption is done till the task finishes. They consider three types of task delay: waiting delay, transmission delay, and execution delay. A DRL-based Policy-Gradient algorithm is used to minimize average task slowdown time. Their evaluation results compared with those of Random and Shortest Execution Time (SET) show that their algorithm performs better in terms of minimization of average task slowdown and resource fairness.

The above review indicates that the policy-gradient methods are mostly used for scheduling and resource allocation by researchers because these algorithms converge faster and are more effective and adaptive in high-dimensional space. Their main disadvantages is that they may converge to local optima, and the evaluation of a policy is typically inefficient and it suffers from high variance.

The actor-Critic algorithm is a hybrid method in which two non-linear neural network function approximators are used. A Critic learns a value function for evaluating the goodness of the taken action, and an Actor is policy-based that receives the state as input and outputs the best action. The critic provides a reinforcing signal to the actor. The training of two networks is separately performed and it uses gradient ascent to update weights. As time passes, the actor learns to take better actions and the critic evaluates those actions. Weight updation happens

at each step as opposed to a policy gradient. DDPG, Proximal Policy Optimization (PPO) [121], and Trust Region Policy Optimization (TRPO) [122] are some examples of variations of the Actor-Critic algorithm.

- **Resource Allocation:** Shreshth Tuli [123] use Asynchronous-Advantage-Actor-Critic (A3C) algorithm [124] for the stochastic dynamic scheduler in edge-cloud environment in which Residual Recurrent Neural Network (R2N2)-based framework is used to exploit temporal patterns in a hybrid environment [125]. They claim that the designed scheduler can quickly adapt to a dynamically changing environment. Their aim is to reduce consumed energy, response time, and cost. They use iFogsim and Cloudsim for simulations. The limitation of their purposed model is the lack of scalability.

Actor-Critic algorithms quickly converge and also solve the problem of variance issue suffered by the policy gradient algorithms. A side-by-side comparison of the dynamic RL and DRL-based scheduling techniques considering the different optimization metrics and evaluation tools is shown in Table XIII. The tick marks indicate the intended criteria while the crosses show the metrics ignored by the researchers.

The focused techniques in the reviewed RL-based literature is presented by check marks in Table XIV.

The above survey shows the initial efforts made by different researchers for solving the task scheduling and allocation problem using RL and DRL-based algorithms. According to Table XIII, most of the proposed algorithms aim to minimize energy consumption along with minimization in turnaround time. However, some other factors as cost, reliability, and security, have been ignored by most of the researchers.

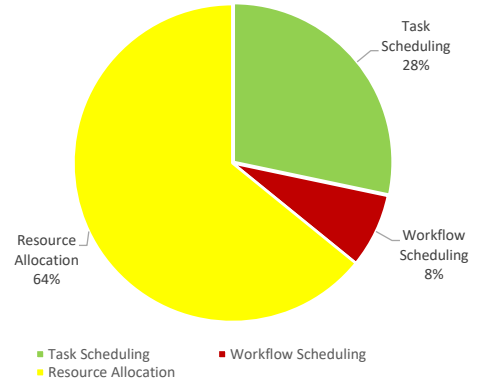



Fig. 8. Focus of Scheduling Techniques.

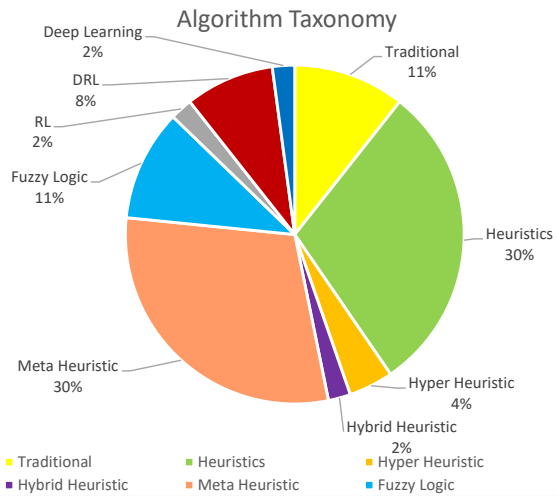



Fig. 9. Classification of Scheduling Algorithms.

## IV. Analytical Discussion

In this section, we provide our analysis of the existing task scheduling techniques in fog computing. The analytical examination is done on the basis of the classification of these algorithms, evaluation environment, tools used, along with the metrics used in these studies.

Fig. 8 shows the percentage of scheduling techniques focused by different researchers. It shows that most of the researchers, i.e., 64% focus on solving resource allocation problems in complex fog computing environments while task scheduling on fog devices is addressed in 28% studies. Only 8% of the studies address the workflow scheduling issue in fog computing.

Fig. 9 shows the classification of various task scheduling and resource allocation algorithms. The analysis suggests that Heuristic and Meta-Heuristic algorithms have widely been used by researchers, i.e., 30% each. Both heuristic and meta-heuristic algorithms are commonly used for solving the scheduling problem because these can find feasible and near-optimal solutions in linear time. For developing self-adaptive scheduling algorithms for online and the uncertain fog computing environment, policy-based DRL algorithms have also been applied by 8% researchers. Traditional algorithms are applied by 11%, fuzzy Logic based algorithms by 10%, and Hyper-heuristic algorithms by 4% researchers. The reinforcement learning, deep learning, and hybrid heuristic algorithms are used by just 2% each.

Fig. 10 shows the evaluation factors that are considered in different approaches. Latency is the most widely used metric, and it is used in 25% of the studies. Among the other metrics, energy consumption has been optimized by 22%, execution time by 9%, QoS by 10%, cost by 8%, network usage by 8%, makespan by 10%, and average slowdown time by 4%, of the studies.

Fig. 11 shows the application environment used by researchers for solving scheduling problems. In 53% studies, fog-cloud environment is considered, while fog environment is considered in 40% of the studies. Edge-fog-cloud is considered in 7% studies.

The software tools used in different studies and their brief introduction are briefly described in table XV, including CloudSim [126], iFogSim [41], WorkflowSim [127], CloudAnalyst [128], SimGrid [129], SimPy [130], and Keras [131].

Fig. 12 shows the simulators that are used for the evaluation of the approaches. As shown in the figure, iFogSim is used in 50% of the simulation studies. Matlab tool is used in around 23% of the evaluation studies. CloudSim tool is used in 14% of the studies. For the rest of the studies, the researchers use WorkflowSim, CloudAnalyst simulators, and C++/Java

TABLE XIII
COMPARISON OF RL AND DRL-BASED SCHEDULING ALGORITHMS.

| Algorithm | Learning | Service Delay | avg slowdown | avg turnaround | Energy Consumption | Execution Time | Evaluation Tool | Reference |
|---|---|---|---|---|---|---|---|---|
| Modified WORN-DEAR | - | × | × | × | ✓ | × | Cooja-Contiki & Experimental | [108] |
| DDQL | Off-Policy | ✓ | × | × | ✓ | ✓ | Keras, Simpy | [113] |
| Policy Gradient | On and Off-Policy | × | ✓ | ✓ | × | × | Experimental | [118] |
| Policy Gradient & DRF | On-Policy | × | ✓ | × | × | × | Simulation(NA) | [119] |
| A3C | On-Policy | × | ✓ | × | ✓ | × | CloudSim | [123] |

TABLE XIV
SCHEDULING TYPE FOCUSED BY RL AND DRL-BASED ALGORITHMS.

| Reference | Environment | Task Scheduling | Resource Allocation | Workflow Scheduling |
|---|---|---|---|---|
| [108] | body-fog-cloud | × | ✓ | × |
| [113] | fog-cloud | × | ✓ | × |
| [118] | fog-cloud | ✓ | × | × |
| [119] | fog | × | ✓ | × |
| [123] | edge-cloud | × | ✓ | × |

TABLE XV
REPRESENTATION OF SOFTWARE TOOLS.

| Software Tools | License | Brief Introduction |
|---|---|---|
| CloudSim | Open-source | CloudSim is the most commonly-used generalized free framework modeling and simulating large-scale cloud infrastructure and its core services. It builds on GridSim. It facilitates the researchers to simulate data centers, virtual machines, cloud, and data center brokers. It allows users to define policies for the allocation of hosts, and host resources, like memory and storage. |
| iFogSim | Open-source | iFogSim is the most popular free simulation toolkit to model and simulate resource management techniques in the fog/cloud and IoT environments. It is based on CloudSim and allows the user to simulate fog infrastructure with millions of fog nodes, sensors, actuators, and IoT services [132] for evaluating resource-management and scheduling policies. It helps the researchers to evaluate various metrics like delay, energy consumption, network usage, and cost [133] |
| WorkflowSim | Open-source | WorkflowSim is a workflow simulator based on CloudSim that provides simulation-based on workflow. It supports researchers simulating different levels of delay and fault in a near-real distributed environment. The simulator facilitates implementing various task scheduling, clustering, and resource management algorithms. |
| CloudAnalyst | Open-source | CloudAnalyst is an open-source simulator built on CloudSim for evaluating the performance of large-scale distributed applications on the cloud. It provides GUI for configuring a geographically distributed system. The evaluation results are presented through graphs and tables. |
| SimGrid | Open-source | SimGrid is a generic framework to simulate distributed applications in large-scale distributed environments like clusters, grids, and cloud. |
| SimPy | Open-source | SimPy is a process-based discrete-event simulation framework that is developed in Python. A Python generator function is used for modeling processes. |
| Keras | Open-source | Keras is a Python library that researchers most commonly use to develop and evaluate deep learning models. |

languages.

The merits and demerits of the techniques used for task scheduling in fog computing are listed in Table XVI.

### A. Lessons Learned from the Survey/ Research Findings

In this survey, we reviewed 46 papers on three major scheduling issues in the fog computing and IoE environment: task scheduling, resource allocation, and workflow scheduling. The techniques used in reviewed studies belong to different domains: traditional, heuristic, hyper-heuristic, hybrid-heuristic, meta-heuristic, and intelligent ones like RL, DRL, and fuzzy-logic based. The survey reveals that most researchers target the resource allocation problem in fog computing, some address task scheduling on fog nodes, while only a few consider workflow scheduling.

The survey indicates that the unpredictable workload, dynamic nature, and complex fog computing and IoE make optimal and self-adaptive resource allocation and scheduling challenges. Most of the approaches need prior information regarding scheduling tasks that are not suitable for a dy-

TABLE XVI
MERITS AND DEMERITS OF TASK SCHEDULING TECHNIQUES.

| Tech. | Reference | Merits | Demerits |
|---|---|---|---|
| Traditional | [41], [42], [44], [45] | • simple, easy to implement<br>• low overhead<br>• deterministic | • not suitable for uncertain and dynamic fog environments<br>• not adaptable |
| Heuristic | [19], [47]–[57], [59], [60] | • simple and cheap to implement<br>• solution in a resonable time | • greedy, so it can easily trap in local optima<br>• Less flexible and scalable |
| Hyper-Heuristic | [62], [63] | • better level of generalization<br>• multi-objective optimization is possible | • difficult to implement<br>• lack learning component |
| Hybrid-Heuristic | [65] | • better results than heuristic algorithms<br>• solution in short time<br>• flexible | • difficult to identify best hybrid solution among all solutions<br>• Not adaptive |
| Meta Heuristic | [70]–[75], [77], [79]–[81], [83]–[87] | • finds near optimal solution in reasonable time<br>• better resource utilization | • long convergence time<br>• mobility factor is not supported |
| Fuzzy Based | [91]–[95] | • suitable for uncertain environment<br>• task prioritization is possible | • longer development time<br>• mathematical descriptions are lacking |
| Q-Learning | [102], [103] | • easy to implement<br>• performs better in uncertain environments | • difficult to generalize<br>• not well suited for complex dynamic fog computing environments |
| Deep Q-Learning | [113] | • performs better than simple Q-learning<br>• Performs well in complex, uncertain/dynamic FC<br>• adaptive in nature | • converges slowly in high-dimensional state-action space. |
| Policy-Gradient | [118], [119], [123] | • convergence is easy<br>• effective in high-dimensional or continuous-space<br>• adaptive | • may converge to local optima<br>• policy evaluation is typically inefficient and suffers from high variance. |

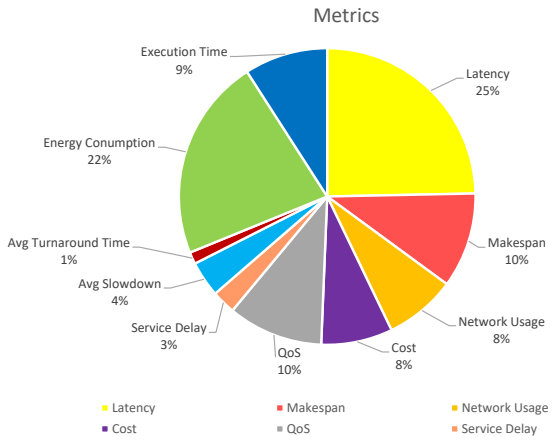



Fig. 10. Metrics used in Scheduling Algorithms.

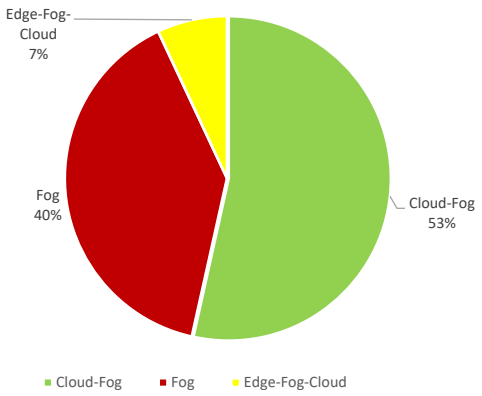



Fig. 11. Application Environments.

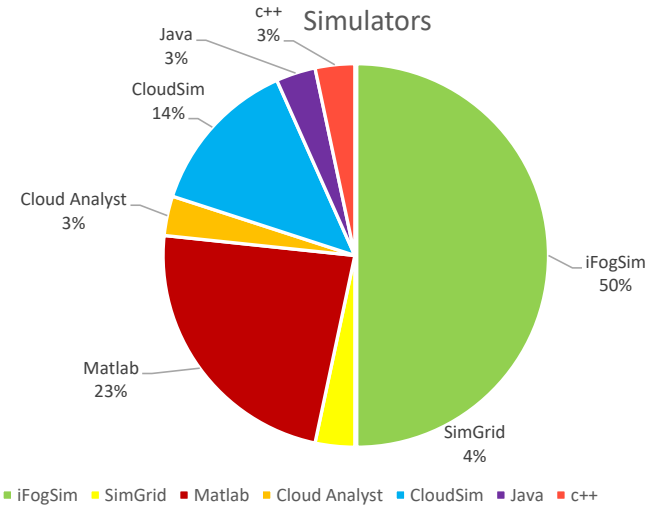



Fig. 12. Simulators used for Task Scheduling Analysis.

namic fog computing environment. Therefore, considering the discussed techniques, fuzzy-based and deep reinforcement learning techniques have great potential to solve these issues.

In different studies, the resource allocation applies on either the fog-cloud layers or only the fog layer and fog computing and IoE. As the cloud and fog environments vary in resource availability and geographical location, this makes the allocation problem more complex. Therefore, the allocation algorithm should fairly, optimally, and efficiently allocate cloud and fog resources to competing IoE requests according to their QoS requirements.

Most of the reviewed studies in the survey are simulation-based, while only a few studies are experimental. In

simulation-based studies, we present a small size simulations scale. Therefore, to assess the performance of these algorithms, extensive simulations are needed. Also, the simulators used in these studies lack the mobility factor as well as the network properties. Therefore, we need simulators that can support these features.

Most studies try to optimize metrics like energy consumption, delay, and network usage while ignoring the security, fault tolerance, and privacy issues. For resource allocation, the trusted node selection is of great importance. Therefore, new algorithms are required that can allocate resources to tasks according to privacy and security needs.

## V. Open Issues and Future Directions

In this section, we discuss the open issues, future challenges, and future research directions for task scheduling in fog computing.

- **Resource Utilization of Fog Node:** The fog devices are resource-constrained in terms of storage, processing, and energy. They receive dynamic workloads from both latency-sensitive and delay-tolerant applications. Therefore, the challenging part is to schedule the unpredictable arrival of tasks on these fog nodes for optimal utilization of available resources.
- **Optimal Resource Allocation:** A big number of tasks are generated by IoE devices that should be optimally allocated to fog nodes to yield faster response time, especially for latency-sensitive applications. As fog computing supports the mobility of fog nodes and IoT devices so, all the resources accessible at one time might be unreachable at any other time. Therefore, this makes resource allocation a challenging task. Long latency for real-time applications, lack of generalization, and quick adaptability of the existing algorithms are the issues that need attention.
- **Parallel Scheduling:** In parallel processing, a task is divided into multiple tasks, and then these sub-tasks are concurrently executed [134]. Dividing tasks into sub-tasks that can reduce delays via distributed computing is another open issue that needs attention.
- **Privacy:** Fog nodes receive a lot of personal data from various fog applications like smart healthcare. Therefore, the privacy of such data is most important for users [135]. Although, some researchers apply privacy-preserving techniques on fog nodes, there is no acceptable authentication solution. The fog nodes are more vulnerable to potential threats that make authentication a challenging issue.
- **Security:** Security is one of the main challenges as fog nodes lack resources and are deployed in unsafe environments, and hence they are easy to attack. Therefore, designing a lightweight, high speed, and reliable safety algorithm is still a challenging task. Currently, only a few researchers focus on security issues in fog computing, and there are some open issues like dynamic authentication, access controls, external attacks, and intrusion detection.
- **Energy Consumption:** As fog devices are energy-constrained because of low-power batteries, energy-aware fog computing is still an open issue that needs to be addressed. Some researchers focus on energy optimization while the appropriate usage of bandwidth in data transfer, energy wastage, and battery drainage issues are still some of the challenges that need attention.
- **Self-Adaptive Scheduling:** Most of the scheduling algorithms lack the learning component that makes self-adaptive resource scheduling a challenge in task scheduling in fog computing. Although, some research studies consider self-adaptive scheduling, all these efforts have only been made on the experimental level. Therefore, task scheduling algorithms are required to optimally schedule tasks generated as a result of unexpected events in a dynamic environment.

Fog computing architecture is distributed as tasks are allocated to heterogeneous fog nodes. Multi-Agent Reinforcement Learning (MARL) is a kind of Reinforcement Learning that can be used for optimal self-adaptive scheduling. Several agents can dynamically learn in multi-agent reinforcement learning after interacting with the environment [136]. In single-agent reinforcement, the learning state is changed due to the actions of the individual agent. In MARL, the state change is concerned with the actions of all agents. MARL for task scheduling has also not been studied by researchers yet.

## VI. Conclusions and Final Remark

Efficient scheduling algorithms can significantly improve the performance of a fog computing and IoE. In this study, we have studied different task scheduling, resource allocation, and workflow scheduling approaches in fog computing and IoE paradigm. We have classified these approaches into eight main categories namely traditional, heuristic, hyper-heuristic, hybrid heuristic, meta-heuristic, fuzzy-based, reinforcement learning, and deep reinforcement learning-based. We have described the advantages, disadvantages, evaluation factors, and simulation environment of every approach. Our analysis indicates that most of the researchers, 30% each, have used heuristic and meta-heuristic algorithms for solving the scheduling problem. Furthermore, 8% researchers have applied policy-based deep reinforcement learning algorithms for solving dynamic task scheduling and allocation problems. Considering the evaluation environment, our analysis shows that 87% of the studies are simulation-based while only 13% studies are experimental. The review also revealed that iFogSim is the most commonly used simulator that has been used in 50% of the studies. Moreover, 23% studies use Matlab for the implementation, and 14% studies use the CloudSim tool for evaluations. Furthermore, the latency has been the most commonly chosen metric for evaluation considered in 25% research studies, energy consumption in 22%, and QoS in 10% studies. The work can be extended by considering the security-aware resource management mechanisms, reliability-based scheduling, privacy, and security architectures in fog computing and IoE.

## References

[1] R. Buyya and A. V. Dastjerdi, *Internet of Things: Principles and paradigms*. Elsevier, 2016.

[2] E. Baccarelli, P. G. V. Naranjo, M. Scarpiniti, M. Shojafar, and J. H. Abawajy, "Fog of everything: Energy-efficient networked computing architectures, research challenges, and a case study," *IEEE access*, vol. 5, pp. 9882–9910, 2017.

[3] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[4] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE communications surveys & tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[5] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation computer systems*, vol. 25, no. 6, pp. 599–616, 2009.

[6] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of internet services and applications*, vol. 1, no. 1, pp. 7–18, 2010.

[7] D. Jo and G. J. Kim, "Iot+ ar: pervasive and augmented environments for "digi-log" shopping experience," *Human-centric Computing and Information Sciences*, vol. 9, no. 1, pp. 1–17, 2019.

[8] A. Chowdhury, G. Karmakar, and J. Kamruzzaman, "The co-evolution of cloud and iot applications: Recent and future trends," in *Handbook of Research on the IoT, Cloud Computing, and Wireless Network Optimization*. IGI Global, 2019, pp. 213–234.

[9] R. Mahmud, F. L. Koch, and R. Buyya, "Cloud-fog interoperability in iot-enabled healthcare solutions," in *Proceedings of the 19th international conference on distributed computing and networking*, 2018, pp. 1–10.

[10] S. Malik and K. Gupta, "Resource scheduling in fog: Taxonomy and related aspects," *Journal of Computational and Theoretical Nanoscience*, vol. 16, no. 10, pp. 4313–4319, 2019.

[11] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[12] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *Journal of Systems Architecture*, 2019.

[13] R. Mahmud, K. Ramamohanarao, and R. Buyya, "Application management in fog computing environments: A taxonomy, review and future directions," *ACM Computing Surveys*, vol. 53, no. 4, pp. 1–43, 2020.

[14] P. G. V. Naranjo, Z. Pooranian, M. Shojafar, M. Conti, and R. Buyya, "Focan: A fog-supported smart city network architecture for management of applications in the internet of everything environments," *Journal of Parallel and Distributed Computing*, vol. 132, pp. 274–283, 2019.

[15] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2015.

[16] M. Ghobaei-Arani, A. Souri, and A. A. Rahmanian, "Resource management approaches in fog computing: A comprehensive review," *Journal of Grid Computing*, pp. 1–42, 2019.

[17] C.-H. Hong and B. Varghese, "Resource management in fog/edge computing: a survey on architectures, infrastructure, and algorithms," *ACM Computing Surveys (CSUR)*, vol. 52, no. 5, pp. 1–37, 2019.

[18] T. Mathew, K. C. Sekaran, and J. Jose, "Study and analysis of various task scheduling algorithms in the cloud computing environment," in *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2014, pp. 658–664.

[19] L. Ni, J. Zhang, C. Jiang, C. Yan, and K. Yu, "Resource allocation strategy in fog computing based on priced timed petri nets," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1216–1228, 2017.

[20] K. Chronaki, A. Rico, M. Casas, M. Moretó, R. M. Badia, E. Ayguadé, J. Labarta, and M. Valero, "Task scheduling techniques for asymmetric multi-core systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 7, pp. 2074–2087, 2017.

[21] A. A. Visheratin, M. Melnik, and D. Nasonov, "Workflow scheduling algorithms for hard-deadline constrained cloud environments," *Procedia Computer Science*, vol. 80, pp. 2098–2106, 2016.

[22] M. Al-Khafajiy, T. Baker, H. Al-Libawy, A. Waraich, C. Chalmers, and O. Alfandi, "Fog computing framework for internet of things applications," in *2018 11th International Conference on Developments in eSystems Engineering (DeSE)*. IEEE, 2018, pp. 71–77.

[23] M. Waqas, Y. Niu, M. Ahmed, Y. Li, D. Jin, and Z. Han, "Mobility-aware fog computing in dynamic environments: Understandings and implementation," *IEEE Access*, vol. 7, pp. 38 867–38 879, 2018.

[24] S. Ghosh, A. Mukherjee, S. K. Ghosh, and R. Buyya, "Mobi-iost: mobility-aware cloud-fog-edge-iot collaborative framework for time-critical applications," *IEEE Transactions on Network Science and Engineering*, 2019.

[25] P. Hosseinioun, M. Kheirabadi, S. R. Kamel Tabbakh, and R. Ghaemi, "atask scheduling approaches in fog computing: A survey," *Transactions on Emerging Telecommunications Technologies*, p. e3792, 2020.

[26] X. Yang and N. Rahmani, "Task scheduling mechanisms in fog computing: review, trends, and perspectives," *Kybernetes*, 2020.

[27] M. R. Alizadeh, V. Khajehvand, A. M. Rahmani, and E. Akbari, "Task scheduling approaches in fog computing: A systematic review," *International Journal of Communication Systems*, p. e4583, 2020.

[28] K. Matrouk and K. Alatoun, "Scheduling algorithms in fog computing: A survey," *International Journal of Networked and Distributed Computing*, 2021.

[29] X. Zhao, Q. Zong, B. Tian, B. Zhang, and M. You, "Fast task allocation for heterogeneous unmanned aerial vehicles through reinforcement learning," *Aerospace Science and Technology*, vol. 92, pp. 588–594, 2019.

[30] A. V. Dastjerdi, H. Gupta, R. N. Calheiros, S. K. Ghosh, and R. Buyya, "Fog computing: Principles, architectures, and applications," in *Internet of Things*. Elsevier, 2016, pp. 61–75.

[31] A. S. Tanenbaum and M. Van Steen, *Distributed systems: principles and paradigms*. Prentice-Hall, 2007.

[32] P. Singh, M. Dutta, and N. Aggarwal, "A review of task scheduling based on meta-heuristics approach in cloud computing," *Knowledge and Information Systems*, vol. 52, no. 1, pp. 1–51, 2017.

[33] M. Kalra and S. Singh, "A review of metaheuristic scheduling techniques in cloud computing," *Egyptian informatics journal*, vol. 16, no. 3, pp. 275–295, 2015.

[34] S. A. Ali and M. Alam, "A relative study of task scheduling algorithms in cloud computing environment," in *2016 2nd International Conference on Contemporary Computing and Informatics (IC3I)*. IEEE, 2016, pp. 105–111.

[35] M. Alodib, "Qos-aware approach to monitor violations of slas in the iot," *Journal of Innovation in Digital Ecosystems*, vol. 3, no. 2, pp. 197–207, 2016.

[36] E. Ibrahim, N. A. El-Bahnasawy, and F. A. Omara, "Task scheduling algorithm in cloud computing environment based on cloud pricing models," in *2016 World Symposium on Computer Applications & Research (WSCAR)*. IEEE, 2016, pp. 65–71.

[37] S. Elmougy, S. Sarhan, and M. Joundy, "A novel hybrid of shortest job first and round robin with dynamic variable quantum time task scheduling technique," *Journal of Cloud Computing*, vol. 6, no. 1, pp. 1–12, 2017.

[38] P. Salot, "A survey of various scheduling algorithm in cloud computing environment," *International Journal of Research in Engineering and Technology*, vol. 2, no. 2, pp. 131–135, 2013.

[39] S. H. H. Madni, M. S. Abd Latiff, M. Abdullahi, S. M. Abdulhamid, and M. J. Usman, "Performance comparison of heuristic algorithms for task scheduling in iaas cloud computing environment," *PloS one*, vol. 12, no. 5, p. e0176321, 2017.

[40] P. B. Galvin, G. Gagne, A. Silberschatz *et al.*, *Operating system concepts*. John Wiley & Sons, 2003.

[41] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.

[42] T. Choudhari, M. Moh, and T.-S. Moh, "Prioritized task scheduling in fog computing," in *ACMSE '18*, 2018.

[43] M. Mtshali, H. Kobo, S. Dlamini, M. Adigun, and P. Mudali, "Multi-objective optimization approach for task scheduling in fog computing," in *2019 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD)*. IEEE, 2019, pp. 1–6.

[44] A. M. Alsmadi, R. M. A. Aloglah, N. J. S. Abu-darwish, A. Al Smadi, M. Alshabanah, D. Alrajhi, H. Alkhaldi, and M. K. Alsmadi, "Fog computing scheduling algorithm for smart city," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 3, pp. 2219–2228, 2021.

[45] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, and M. Parashar, "Mobility-aware application scheduling in fog computing," *IEEE Cloud Computing*, vol. 4, no. 2, pp. 26–35, 2017.

[46] N. Soltani, B. Soleimani, and B. Barekatain, "Heuristic algorithms for task scheduling in cloud computing: a survey," *International Journal of Computer Network and Information Security*, vol. 9, no. 8, p. 16, 2017.

[47] B. Jamil, M. Shojafar, I. Ahmed, A. Ullah, K. Munir, and H. Ijaz, "A job scheduling algorithm for delay and performance optimization in fog computing," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 7, p. e5581, 2020.

[48] D. Hoang and T. D. Dang, "Fbrc: Optimization of task scheduling in fog-based region and cloud," in *Trustcom/BigDataSE/ICESS, 2017 IEEE*. IEEE, 2017, pp. 1109–1114.

[49] Y. Yang, K. Wang, G. Zhang, X. Chen, X. Luo, and M.-T. Zhou, "Meets: Maximal energy efficient task scheduling in homogeneous fog networks," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 4076–4087, 2018.

[50] F. Hoseiny, S. Azizi, M. Shojafar, and R. Tafazolli, "Joint qos-aware and cost-efficient task scheduling for fog-cloud resources in a volunteer computing system," *ACM Transactions on Internet Technology*, pp. 1–21, 2021.

[51] G. L. Stavrinides and H. D. Karatza, "A hybrid approach to scheduling real-time iot workflows in fog and cloud environments," *Multimedia Tools and Applications*, vol. 78, no. 17, pp. 24 639–24 655, 2019.

[52] N. Auluck, A. Azim, and K. Fizza, "Improving the schedulability of real-time tasks using fog computing," *IEEE Transactions on Services Computing*, 2019.

[53] R. M. Abdelmoneem, A. Benslimane, and E. Shaaban, "Mobility-aware task scheduling in cloud-fog iot-based healthcare architectures," *Computer Networks*, p. 107348, 2020.

[54] D. Zeng, L. Gu, S. Guo, Z. Cheng, and S. Yu, "Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system," *IEEE Transactions on Computers*, vol. 65, no. 12, pp. 3702–3712, 2016.

[55] Z. Pooranian, M. Shojafar, P. G. V. Naranjo, L. Chiaraviglio, and M. Conti, "A novel distributed fog-based networked architecture to preserve energy in fog data centers," in *2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*. IEEE, 2017, pp. 604–609.

[56] G. Zhang, F. Shen, N. Chen, P. Zhu, X. Dai, and Y. Yang, "Dots: Delay-optimal task scheduling among voluntary nodes in fog networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3533–3544, 2018.

[57] Y. Yang, S. Zhao, W. Zhang, Y. Chen, X. Luo, and J. Wang, "Debts: Delay energy balanced task scheduling in homogeneous fog networks," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2094–2106, 2018.

[58] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

[59] X.-Q. Pham and E.-N. Huh, "Towards task scheduling in a cloud-fog computing system," in *Network Operations and Management Symposium (APNOMS), 2016 18th Asia-Pacific*. IEEE, 2016, pp. 1–4.

[60] J. C. Guevara and N. L. da Fonseca, "Task scheduling in cloud-fog computing systems," *Peer-to-Peer Networking and Applications*, pp. 1–16, 2021.

[61] J. H. Drake, A. Kheiri, E. Özcan, and E. K. Burke, "Recent advances in selection hyper-heuristics," *European Journal of Operational Research*, vol. 285, no. 2, pp. 405–428, 2020.

[62] L. Liu, D. Qi, N. Zhou, and Y. Wu, "A task scheduling algorithm based on classification mining in fog computing environment," *Wireless Communications and Mobile Computing*, vol. 2018, 2018.

[63] S. Kabirzadeh, D. Rahbari, and M. Nickray, "A hyper heuristic algorithm for scheduling of fog networks," in *2017 21st Conference of Open Innovations Association (FRUCT)*. IEEE, 2017, pp. 148–155.

[64] C.-W. Tsai, W.-C. Huang, M.-H. Chiang, M.-C. Chiang, and C.-S. Yang, "A hyper-heuristic scheduling algorithm for cloud," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 236–250, 2014.

[65] J. Wang and D. Li, "Task scheduling based on a hybrid heuristic algorithm for smart production line with fog computing," *Sensors*, vol. 19, no. 5, p. 1023, 2019.

[66] E. H. Houssein, A. G. Gad, Y. M. Wazery, and P. N. Suganthan, "Task scheduling in cloud computing based on meta-heuristics: Review, taxonomy, open challenges, and future trends," *Swarm and Evolutionary Computation*, p. 100841, 2021.

[67] M. Abdel-Basset, L. Abdel-Fatah, and A. K. Sangaiah, "Metaheuristic algorithms: A comprehensive review," in *Computational intelligence for multimedia big data on the cloud with engineering applications*. Elsevier, 2018, pp. 185–231.

[68] C.-W. Tsai and J. J. Rodrigues, "Metaheuristic scheduling for cloud: A survey," *IEEE Systems Journal*, vol. 8, no. 1, pp. 279–291, 2013.

[69] X.-S. Yang, S. F. Chien, and T. O. Ting, "Computational intelligence and metaheuristic algorithms with applications," 2014.

[70] Q. Liu, Y. Wei, S. Leng, and Y. Chen, "Task scheduling in fog enabled internet of things for smart cities," in *2017 IEEE 17th International Conference on Communication Technology (ICCT)*. IEEE, 2017, pp. 975–980.

[71] Y. Wang, C. Guo, and J. Yu, "Immune scheduling network based method for task scheduling in decentralized fog computing," *Wireless Communications and Mobile Computing*, vol. 2018, 2018.

[72] B. M. Nguyen, H. Thi Thanh Binh, B. Do Son *et al.*, "Evolutionary algorithms to optimize task scheduling problem for the iot based bag-of-tasks application in cloud–fog computing environment," *Applied Sciences*, vol. 9, no. 9, p. 1730, 2019.

[73] S. Bitam, S. Zeadally, and A. Mellouk, "Fog computing job scheduling optimization based on bees swarm," *Enterprise Information Systems*, vol. 12, no. 4, pp. 373–397, 2018.

[74] D. Rahbari and M. Nickray, "Low-latency and energy-efficient scheduling in fog-based iot applications," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 27, no. 2, pp. 1406–1427, 2019.

[75] K. N. Jayasena and B. Thisarasinghe, "Optimized task scheduling on fog computing environment using meta heuristic algorithms," in *2019 IEEE International Conference on Smart Cloud (SmartCloud)*. IEEE, 2019, pp. 53–58.

[76] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in engineering software*, vol. 95, pp. 51–67, 2016.

[77] J. Xu, Z. Hao, R. Zhang, and X. Sun, "A method based on the combination of laxity and ant colony system for cloud-fog task scheduling," *IEEE Access*, vol. 7, pp. 116 218–116 226, 2019.

[78] H. Topcuoglu, S. Hariri, and M.-y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE transactions on parallel and distributed systems*, vol. 13, no. 3, pp. 260–274, 2002.

[79] Y. Sun, F. Lin, and H. Xu, "Multi-objective optimization of resource scheduling in fog computing using an improved nsga-ii," *Wireless Personal Communications*, vol. 102, no. 2, pp. 1369–1385, 2018.

[80] R. O. Aburukba, M. AliKarrar, T. Landolsi, and K. El-Fakih, "Scheduling internet of things requests to minimize latency in hybrid fog–cloud computing," *Future Generation Computer Systems*, vol. 111, pp. 539–551, 2020.

[81] M. Ghobaei-Arani, A. Souri, F. Safara, and M. Norouzi, "An efficient task scheduling approach using moth-flame optimization algorithm for cyber-physical system applications in fog computing," *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 2, p. e3770, 2020.

[82] S. Mirjalili, "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm," *Knowledge-based systems*, vol. 89, pp. 228–249, 2015.

[83] S. Wang, T. Zhao, and S. Pang, "Task scheduling algorithm based on improved firework algorithm in fog computing," *IEEE Access*, vol. 8, pp. 32 385–32 394, 2020.

[84] M. Yang, H. Ma, S. Wei, Y. Zeng, Y. Chen, and Y. Hu, "A multi-objective task scheduling method for fog computing in cyber-physical-social services," *IEEE Access*, vol. 8, pp. 65 085–65 095, 2020.

[85] F. Hoseiny, S. Azizi, M. Shojafar, F. Ahmadizar, and R. Tafazolli, "Pga: A priority-aware genetic algorithm for task scheduling in heterogeneous fog-cloud computing," in *Proc. IEEE INFOCOM ICCN'21*, 2021, pp. 1–6.

[86] N. Potu, C. Jatoth, and P. Parvataneni, "Optimizing resource scheduling based on extended particle swarm optimization in fog computing environments," *Concurrency and Computation: Practice and Experience*, p. e6163, 2021.

[87] R. Xu, Y. Wang, Y. Cheng, Y. Zhu, Y. Xie, A. S. Sani, and D. Yuan, "Improved particle swarm optimization based workflow scheduling in cloud-fog environment," in *International Conference on Business Process Management*. Springer, 2018, pp. 337–347.

[88] L. A. Zadeh, G. J. Klir, and B. Yuan, *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers*. World Scientific, 1996, vol. 6.

[89] L. A. Zadeh, "Soft computing and fuzzy logic," in *Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems: Selected Papers by Lotfi a Zadeh*. World Scientific, 1996, pp. 796–804.

[90] M. I. Tariq, S. Tayyaba, M. W. Ashraf, M. Imran, E. Pricop, O. Cangea, N. Paraschiv, and N. Ali Mian, "An analysis of the application of fuzzy logic in cloud computing," *Journal of Intelligent & Fuzzy Systems*, no. Preprint, pp. 1–15, 2020.

[91] M. A. Benblidia, B. Brik, L. Merghem-Boulahia, and M. Esseghir, "Ranking fog nodes for tasks scheduling in fog-cloud environments: A fuzzy logic approach," in *2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*. IEEE, 2019, pp. 1451–1457.

[92] G. Li, Y. Liu, J. Wu, D. Lin, and S. Zhao, "Methods of resource scheduling based on optimized fuzzy clustering in fog computing," *Sensors*, vol. 19, no. 9, p. 2122, 2019.

[93] S. Javanmardi, M. Shojafar, V. Persico, and A. Pescapè, "Fpfts: A joint fuzzy particle swarm optimization mobility-aware approach to fog task scheduling algorithm for internet of things devices," *Software: Practice and Experience*, 2020.

[94] C.-g. Wu, W. Li, L. Wang, and A. Y. Zomaya, "An evolutionary fuzzy scheduler for multi-objective resource allocation in fog computing," *Future Generation Computer Systems*, vol. 117, pp. 498–509, 2021.

[95] H. S. Ali, R. R. Rout, P. Parimi, and S. K. Das, "Real-time task scheduling in fog-cloud computing framework for iot applications: A fuzzy logic based approach," in *2021 International Conference on COMmunication Systems & NETworkS (COMSNETS)*. IEEE, 2021, pp. 556–564.

[96] Y. M. Jiménez, "A generic multi-agent reinforcement learning approach for scheduling problems," *PhD, Vrije Universiteit Brussel*, p. 128, 2012.

[97] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[98] P. H. Valente Klaine, "Self-organization for 5g and beyond mobile networks using reinforcement learning," Ph.D. dissertation, University of Glasgow, 2019.

[99] Y. C. F. Reyna, Y. M. Jiménez, J. M. B. Cabrera, and B. M. M. Hernández, "A reinforcement learning approach for scheduling problems," *Investigación Operacional*, vol. 36, no. 3, pp. 225–231, 2015.

[100] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[101] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[102] A. I. Orhean, F. Pop, and I. Raicu, "New scheduling approach using reinforcement learning for heterogeneous distributed systems," *Journal of Parallel and Distributed Computing*, vol. 117, pp. 292–302, 2018.

[103] X. Liu, Z. Qin, and Y. Gao, "Resource allocation for edge computing in iot networks via reinforcement learning," *arXiv preprint arXiv:1903.01856*, 2019.

[104] Q. Wu, Z. Wu, Y. Zhuang, and Y. Cheng, "Adaptive dag tasks scheduling with deep reinforcement learning," in *International Conference on Algorithms and Architectures for Parallel Processing*. Springer, 2018, pp. 477–490.

[105] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1, no. 2.

[106] O. Faust, Y. Hagiwara, T. J. Hong, O. S. Lih, and U. R. Acharya, "Deep learning for healthcare applications based on physiological signals: A review," *Computer methods and programs in biomedicine*, vol. 161, pp. 1–13, 2018.

[107] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[108] S. Amudha and M. Murali, "Deep learning based energy efficient novel scheduling algorithms for body-fog-cloud in smart hospital," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–20, 2020.

[109] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with cooja," in *Proceedings. 2006 31st IEEE Conference on Local Computer Networks*. IEEE, 2006, pp. 641–648.

[110] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," *arXiv preprint arXiv:1708.05866*, 2017.

[111] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[112] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1995–2003.

[113] P. Gazori, D. Rahbari, and M. Nickray, "Saving time and cost on the scheduling of fog-based iot applications using deep reinforcement learning approach," *Future Generation Computer Systems*, 2019.

[114] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances in neural information processing systems*, 2000, pp. 1008–1014.

[115] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[116] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*. ACM, 2016, pp. 50–56.

[117] W. Chen, Y. Xu, and X. Wu, "Deep reinforcement learning for multi-resource multi-machine job scheduling," *arXiv preprint arXiv:1711.07440*, 2017.

[118] Y. Ye, X. Ren, J. Wang, L. Xu, W. Guo, W. Huang, and W. Tian, "A new approach for resource scheduling with deep reinforcement learning," *arXiv preprint arXiv:1806.08122*, 2018.

[119] S. Bian, X. Huang, and Z. Shao, "Online task scheduling for fog computing with multi-resource fairness," in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*. IEEE, 2019, pp. 1–5.

[120] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: Fair allocation of multiple resource types." in *Nsdi*, vol. 11, no. 2011, 2011, pp. 24–24.

[121] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[122] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[123] S. Tuli, S. Ilager, K. Ramamohanarao, and R. Buyya, "Dynamic scheduling for stochastic edge-cloud computing environments using a3c learning and residual recurrent neural networks," *IEEE Transactions on Mobile Computing*, 2020.

[124] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016, pp. 1928–1937.

[125] B. Yue, J. Fu, and J. Liang, "Residual recurrent neural networks for learning sequential representations," *Information*, vol. 9, no. 3, p. 56, 2018.

[126] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011.

[127] W. Chen and E. Deelman, "Workflowsim: A toolkit for simulating scientific workflows in distributed environments," in *2012 IEEE 8th international conference on E-science*. IEEE, 2012, pp. 1–8.

[128] B. Wickremasinghe, R. N. Calheiros, and R. Buyya, "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*. IEEE, 2010, pp. 446–452.

[129] H. Casanova, A. Legrand, and M. Quinson, "Simgrid: A generic framework for large-scale distributed experiments," in *Tenth International Conference on Computer Modeling and Simulation (uksim 2008)*. IEEE, 2008, pp. 126–131.

[130] N. Matloff, "Introduction to discrete-event simulation and the simpy language," *Davis, CA. Dept of Computer Science. University of California at Davis. Retrieved on August*, vol. 2, no. 2009, pp. 1–33, 2008.

[131] A. Gulli and S. Pal, *Deep learning with Keras*. Packt Publishing Ltd, 2017.

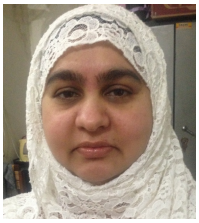
[132] R. Mahmud and R. Buyya, "Modelling and simulation of fog and edge computing environments using ifogsim toolkit," *Fog and edge computing: Principles and paradigms*, pp. 1–35, 2019.

[133] D. P. Abreu, K. Velasquez, M. Curado, and E. Monteiro, "A comparative analysis of simulators for the cloud to fog continuum," *Simulation Modelling Practice and Theory*, vol. 101, p. 102029, 2020.

[134] B. Barney, *Introduction to Parallel Computing*. USA: Lawrence Livermore National Laboratory, 2015.

[135] A. A. Laghari, A. K. Jumani, and R. A. Laghari, "Review and state of art of fog computing," *Archives of Computational Methods in Engineering*, pp. 1–13, 2021.

[136] D. Kim, S. Moon, D. Hostallero, W. J. Kang, T. Lee, K. Son, and Y. Yi, "Learning to schedule communication in multi-agent reinforcement learning," *arXiv preprint arXiv:1902.01554*, 2019.



**Bushra Jamil** is doing her Ph.D. degree from University of Sargodha, Sargodha, Pakistan. She received the M.Sc in Computer Science degree from Quaid-e-Azam University Islamabad, Pakistan. She received the MS degree from University of Sargodha (UOS), Pakistan. She is working as a Lecturer in Department of Computer Science and Information Technology, University of Sargodha (UOS), Sargodha, Pakistan. Her areas of research include Fog Computing, Machine Learning and Deep Learning.

**Humaira Ijaz** has done her Ph.D.from University of Innsbruck, Austria in 2016. She received her MSc. degree from (PUCIT) University of Punjab, Lahore, Pakistan. She is working as an Assistant Professor in Department of Computer Science and Information Technology, University of Sargodha (UOS), Sargodha, Pakistan. Her areas of research include Fog Computing, Cloud Computing and Machine learning.

**Mohammad Shojafar (M'17-SM'19)** is a Senior Lecturer (Associate Professor) in the network security and an Intel Innovator, Profassional ACM member, and a Marie Curie Alumni, working in the 5G & 6G Innovation Centre (5GIC/6GIC) at the University of Surrey, UK. Before joining 5GIC/6GIC, he was a Senior Researcher and a Marie Curie Fellow in the SPRITZ Security and Privacy Research group at the University of Padua, Italy. Also, he was CNIT Senior Researcher at the University of Rome Tor Vergata contributed to 5G PPP European H2020 "SUPERFLUIDITY" project. Dr. Mohammad was a PI of PRISENODE project, a 275k euro Horizon 2020 Marie Curie global fellowship project in the areas of Fog/Cloud security collaborating at the University of Padua. He received his Ph.D. degree from Sapienza University of Rome, Rome, Italy, in 2016 with an "Excellent" degree. He is an Associate Editor in *IEEE Transactions on Consumer Electronics*, *IEEE Systems Journal* and Computer Networks. For additional information: http://mshojafar.com

**Kashif Munir** received his Ph.D. degree from University of Innsbruck, Austria in 2009. He was a post-doctoral researcher at IMT Atlantique (formerly known as Telecom Bretagne), France from February 2011 to December 2012. He is working as a Head of Department and Associate Professor in the department of Computer Science at National University of Computer and Emerging Sciences (NUCES), Islamabad, Pakistan. His areas of research include admission and congestion control, quality of service for bulk data transfers, performance modeling of computer and communication systems, high-performance computing, and mobility cost analysis. He is a reviewer of IEEE transactions on Communications, Computer Networks (Elsevier), Telecommunication Systems (Springer), Cluster Computing (Springer), Concurrency and Computation (Wiley), Journal of Network and Computer Applications (Elsevier), International Journal of Computer Mathematics, and Journal of King Saud University-Computer and Information Sciences. He is an author of numerous peer-reviewed conference and journal publications.

**Rajkumar Buyya (F'17)** is a Redmond Barry Distinguished Professor and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He has authored over 750 publications and seven text books including "Mastering Cloud Computing" published by McGraw Hill, China Machine Press, and Morgan Kaufmann for Indian, Chinese and international markets respectively. Dr. Buyya is one of the highly cited authors in computer science and software engineering worldwide (h-index = 146, g-index = 323, 113K+ citations). He served as founding Editor-in-Chief of the IEEE Transactions on Cloud Computing. He is currently serving as Editor-in-Chief of Software: Practice and Experience. For additional information: www.buyya.com