IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 30 November 2023, accepted 13 December 2023, date of publication 18 December 2023,
date of current version 22 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3343877

TOPICAL REVIEW

A Review on Task Scheduling Techniques in Cloud and Fog Computing: Taxonomy, Tools, Open Issues, Challenges, and Future Directions

ZULFIQAR ALI KHAN[1], IZZATDIN ABDUL AZIZ[1], NURUL AIDA BT OSMAN[1], AND ISRAR ULLAH[2]

[1]Department of Computer and Information Sciences, Universiti Teknologi PETRONAS, Seri Iskandar 32610, Malaysia
[2]Department of Computer Science and Information Technology, Virtual University of Pakistan, Lahore 25100, Pakistan

Corresponding author: Zulfiqar Ali Khan (zulfiqar_22005381@utp.edu.my)

This work was supported by Yayasan Universiti Teknologi PETRONAS (UTP), Malaysia, with titled "Predicting Missing Values in Big Upstream Oil and Gas Industrial Dataset Using Enhanced Evolved Bat Algorithm and Support Vector Regression" under the Center for Research in Data Science (CerDaS), UTP, under Grant 015LC0-353.

ABSTRACT Efficient task scheduling on the cloud is critical for optimal utilization of resources in data centers. It became even more challenging with the emergence of 5G and IoT applications that generate massive number of tasks with stringent latency requirements. This gives birth to fog/edge computing - a complementary layer to cloud. Tasks latency in fog computing can be reduced as processing in the network is done closer to the end devices and users, but every task cannot be scheduled in the fog due to limited resources availability. Conventional scheduling algorithms often fail to exploit the heterogeneous resources; therefore, specially designed and well-tuned scheduling algorithms are desired for achieving better quality of service. In this study, the state-of-the-art task scheduling algorithms in the cloud and fog environments are investigated in a diverse set of dimensions. Among the relevant studies published between 2018-2022 and indexed in the Web-of-Science (WOS), SCOPUS, and Google Scholar databases, eighteen studies are selected for both the cloud and fog domains from WOS and Scopus, while seventeen studies are chosen for both the cloud and fog domains from Google Scholar. Thus, a total of 106 studies are included in this survey for the detail investigation. The scheduling algorithms are broadly classified into three categories such as heuristic, meta-heuristic, and hybrid meta-heuristic followed by detailed critical analysis. It has been observed that most of the scheduling algorithms are dynamic and non-preemptive in nature, while the higher fraction of the tasks is independent in comparison to bag of tasks and workflows. Similarly, 97% of the studies focus on multiple objectives and 68% of the techniques are non-deterministic. Further, a total of twenty different scheduling objectives are identified with makespan, resource utilization, delay, load balancing, and energy consumption as the most significant metrics. The evaluation methods including simulations (51%), real experiments (4%), analytical equations (2%), and datasets (43%) etc. are surveyed. At the end, the open issues, challenges, and future directions are argued.

INDEX TERMS Cloud computing, fog computing, task scheduling, workflow, Internet of Things (IoT).

I. INTRODUCTION

Cloud computing offers different kinds of scalable resources to end users based on pay-per-use basis by providing cost savings, security, flexibility, and automatic software updates etc. The users can concentrate on their personal and business objectives without bothering about the infrastructure and supporting services as the vendor is responsible for managing it. There are three service models and four deployment models provided by the cloud providers [1]. Though it is

The associate editor coordinating the review of this manuscript and approving it for publication was Zhiwu Li.

presumed that there is an unlimited number of resources on the cloud, in fact it is not true. The service providers strive their best to reduce the hardware and software expenses for maximizing profits. Cloud computing offers a platform to run tasks on rented Virtual Machines (VMs). Being a pay per use model, researchers are trying to come up with efficient and cost-effective solutions for task execution on VMs. The incoming tasks are assigned to VMs that can fulfill the associated Quality of Service (QoS) requirements. Therefore, optimal usage of the resources is crucial.

With a variety of solutions for data management and improvements in transmission technologies, it became imminent to use the cloud infrastructure for storing the gigantic data from various devices/machines and data analytics. In addition to the cloud, the cost efficiency of Internet of Things (IoT) technologies resulted in the production of billions of devices worldwide. The IoT devices capture and transmit the data from its environment to store on the cloud.

Over the years, the IoT devices have become intelligent, but resource hungry at the same time. The cloud serves the needs of these devices, but at the cost of long transmission delays. Intelligent devices executing near real-time tasks need the resources on urgent basis and cannot tolerate delays. To cater the needs of such devices, Fog Computing has emerged. Instead of a cloud competitor, fog computing complements it by reducing the transmission delays and offer services at the network edge as shown in figure1. Fog devices offer computation and storage services to the IoT devices. But substantial number of requests and limited resources at the fog nodes need optimum scheduling algorithm. Though the fog infrastructure is closer to the end users, it has limited computing and storages resources. Therefore, task scheduling is equally important in fog computing to provide services to the end users in a resource constraint environment with substantial number of user requests.

Cloud computing solves the problems of resource availability, while fog computing complements the cloud paradigm addressing the delay bottlenecks for deadline constraint tasks.

A. MOTIVATION AND RESEARCH QUESTIONS

There are ample Systematic Literature Reviews (SLR) for task scheduling on the cloud in comparison with a limited number of such studies at the fog. In [2], an SLR on the fog task scheduling algorithms provided the study motivation, analysis, taxonomy, classification, and comparison of the existing works with relevant discussion. The open issues and future directions were also highlighted. But the algorithmic approaches, nature of algorithms, datasets, disparate evaluation methods, and comparison of simulation tools were not included.

For cloud computing, a task scheduling SLR categorized different scheduling approaches followed by a taxonomy and classification. The pros and cons of the scheduling approaches were presented in addition to the challenges and

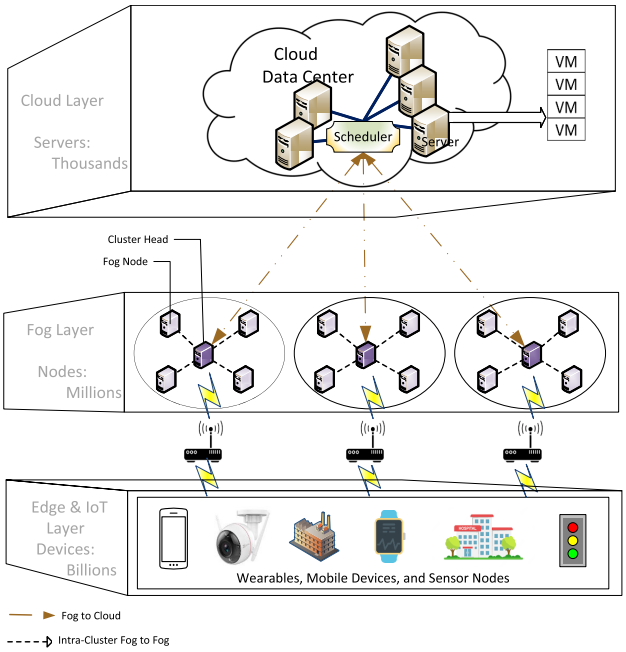

FIGURE 1. Three layered architecture.

open issues [3]. Nonetheless, the algorithmic approaches, datasets, evaluation methods, and comparison of the simulation tools were not covered. A survey on cloud task scheduling reported different approaches from 2005 to 2018 by reviewing 65 research papers [4]. The algorithms were organized based on the application and parameters in addition to the future directions. But the study motivation, research questions, algorithms approaches, taxonomy, nature of the algorithms, datasets, challenges, different evaluation methods, and comparison of simulation tools were not catered.

Another SLR for fog task scheduling approaches reviewed 56 studies after presenting inclusion/exclusion criteria in [5]. The studies were summarized by mentioning their strengths/weaknesses. Further the research gaps were identified in the existing solutions along with future directions. However, evaluation methods were not highlighted categorically. A comprehensive review on task and workflow scheduling algorithms in cloud presented the taxonomy and classification of multi-objective frameworks in [6]. The results of the selected studies were compared by presenting the issues and open research areas. But the paper selection procedure, the time span of the selected studies, nature of algorithms, evaluation techniques, and comparison of simulation tools were not part of the review.

In [7], the meta-heuristic algorithms for task scheduling on the cloud reviewed the categorization of different techniques, taxonomy, and the advantages and disadvantages of different algorithms. Open issues and future trends were highlighted along with a comparison of frequently used simulation tools. Yet the algorithmic approaches and analysis of datasets are ignored in the study. Another review on fog task scheduling investigated and categorized

TABLE 1. Comparison of the cloud and fog review studies.

Serial No.	Past Review Papers	Year (Published)	Duration	No. of Included Studies	Motivation and Research Questions	Algorithmic Approaches	Paper Selection Methodology	Taxonomy	Nature of Algorithms	Datasets	Challenges	Evaluation Methods	Comparison of Simulation Tools
1	[2]	2019	2012-2020	100	✓	×	✓	✓	×	×	✓	×	×
2	[3]	2019	2011-2018	63	✓	×	✓	✓	✓	×	✓	×	×
3	[4]	2019	2005-2018	65	×	×	✓	×	×	×	×	×	×
4	[5]	2020	2015-2020	56	✓	✓	✓	✓	✓	✓	✓	×	✓
5	[6]	2020	-	-	✓	✓	×	✓	×	✓	✓	×	×
6	[7]	2021	2013-2020	71	✓	×	✓	✓	✓	×	✓	✓	✓
7	[8]	2021	2012-2018	15	✓	×	✓	✓	×	×	✓	×	×
8	[9]	2022	-	-	✓	✓	×	✓	×	×	✓	×	×
	Our Work		2018-2022	106	✓	✓	✓	✓	✓	✓	✓	✓	✓

the algorithms as heuristics and meta-heuristics [8]. The existing algorithms, challenges, and future directions were emphasized. Nevertheless, only 15 studies were part of the review and algorithmic approaches, nature of algorithms, datasets, different evaluation methods, and comparison of simulation tools were not covered in this study. A recent survey on fog computing and internet of everything presented the different optimization metrics followed by classifying the existing scheduling techniques [9]. The studies were analyzed for identifying open issues and possible future approaches. But neither the paper selection process nor the duration of the studies was mentioned. Further, the nature for the algorithms, datasets, different evaluation methods, and comparison of the simulation tools were not part of the survey.

There is no comprehensive SLR up to our best knowledge that covers the task scheduling on both the cloud and fog computing till date. The existing review studies lack some aspects like algorithmic approaches, nature of algorithms, datasets, evaluation methods, and comparison of simulation tools etc. as given in Table 1, therefore a systematic study covering the different dimensions of task scheduling on the cloud and fog computing is presented in this article.

The study attempts to answer the following research questions:

1) What are the most commonly used techniques for task scheduling on the cloud and fog in the past five years?
2) Which metrics are considered by the task scheduling algorithms?
3) Which evaluation methods are applied to verify the efficacy of the scheduling algorithms?
4) Which tools/simulators are widely used for implementing and evaluation of the cloud and fog task scheduling?
5) How many studies discussed the time complexity of their proposed techniques?
6) How many studies performed real experiments to validate their proposed techniques?
7) Which research gaps need more exploration in task scheduling approaches?
8) What are the current issues, impediments and future directions in the cloud and fog task scheduling?

B. OUR CONTRIBUTIONS AND ORGANIZATION OF PAPER:

The following are this SLR's main contributions:

1) A motivation for the study and research questions.
2) The taxonomy and classification of task scheduling algorithms on the cloud and fog computing.
3) The study of the proposed algorithms based on resource mapping and preemptive/non-preemptive algorithms.
4) Identifying the types of tasks, number of objectives, and algorithmic approaches used for task scheduling on the cloud and fog.
5) A comprehensive analysis of heuristics, meta-heuristics, and hybrid meta-heuristic algorithms with critical insight.
6) The assessment of all task scheduling objectives discovered in the selected articles on the cloud and fog.
7) The discussion on various evaluation methods including simulation tools, datasets, and real experiments etc.
8) Discussing the open issues, challenges, and future directions.

The remaining sections are categorized as follows: Section II describes the paper selection procedure followed by the computing environments in Section III. Resource mapping, nature of scheduling algorithms, types of tasks, and number of objectives are presented in Sections IV to VII respectively. Algorithmic approaches, categories of solutions, scheduling objectives, and evaluation techniques are given in Sections VIII to XI. Open issues, challenges, and future directions are stated in Section XII-XIII.

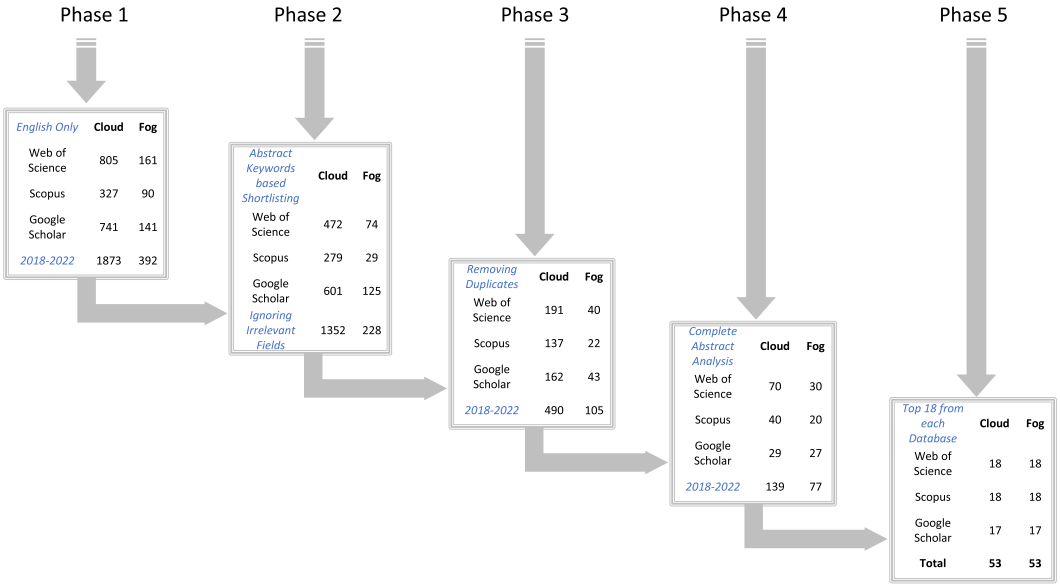

FIGURE 2. Shortlisting of articles.

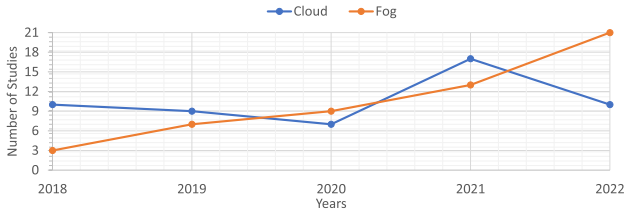

FIGURE 3. Year-wise paper distribution.

C. PAPER SELECTION PROCEDURE

The duration of last five years (2018 to 2022) is chosen for this study based on the keywords of (schedule OR scheduling) AND (cloud) and (schedule OR scheduling) AND (fog) in phase one. The papers in other languages besides English have been ignored. In the second phase, for Web-of-Science (WOS) and Scopus, shortlisting is performed based on the abstracts considering the keywords of (task OR workflow) and removed the papers related to irrelevant fields. In the case of Google Scholar, the edge computing related papers have been excluded. The duplicates are removed in the third phase, while the abstracts are analyzed for further shortlisting in phase 4. Finally, the top 18 papers are selected from WOS and Scopus, and 17 papers from Google Scholar in phase five. The complete process of paper selection is shown in Figure 2.

The year wise distribution of fifty-three papers each for task scheduling on the cloud and fog is depicted in Figure 3 and the trend clearly indicates growing interest in this domain. For task scheduling on the cloud and fog, the maximum number of papers are published in 2021 and 2022 respectively.

II. ENVIRONMENT

The scope of this study spans over the cloud and fog computing paradigms due to a wide range of similarities in scheduling of tasks. Yet there are several factors that make the task scheduling different on the two platforms altogether, but such differences are highlighted wherever required in the article. Task scheduling is equally critical in both the cloud and fog computing but targets dissimilar objective functions. Resource utilization, load balancing, and SLA violations are important on the cloud, while delay, energy consumption, and meeting the deadlines have the same level of importance on the fog. Cloud being a resource rich platform has a lot of resources making use of virtualization technology [10]. One server machine hosts hundreds of VMs that make efficient use of the physical machine's resources, thus providing cost efficient services to multiple concurrent users. But these servers are distant from the end users and suffer from transmission delays, thus not feasible for delay-intolerant applications. Fog computing on the other hand, is a decentralized system residing next to the network edge- closer to end-devices and users. It provides services for delay intolerant tasks but possess limited resources [5]. Besides the cloud's high pricing, the data escalates at a rapid pace on the cloud that makes it difficult to provide real time response. Therefore, fog computing as complementary architecture handles delay intolerant tasks in real time and lowers the overall costs. So, the cloud resources are provisioned only for long term data management and analytics.

Figure 4 presents the complete taxonomy of task scheduling on the cloud and fog.

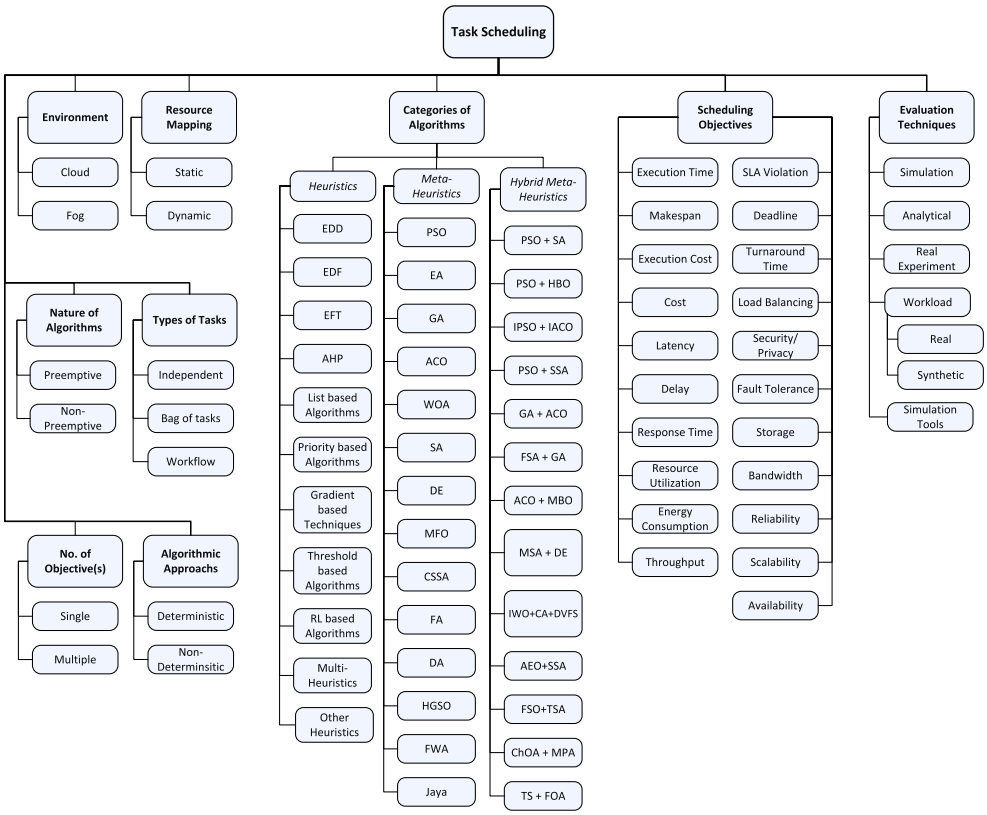

FIGURE 4. Taxonomy of task scheduling on the cloud and fog.

III. RESOURCE MAPPING

The assignment of computing resources to the incoming tasks is called resource mapping. During task scheduling, the resources can be mapped in two ways, static and dynamic.

A static scheduler has a predefined number of tasks in hand; thus, it utilizes the given information to come up with an accurate schedule providing minimum scheduling overhead [11]. For example, Round Robin (RM), Shortest Job First (SJF), and First Come First Serve (FCFS). The static algorithms explored in this study are [12], [13], [14], [15], [16], and [17]. On the contrary, a dynamic scheduler does not know in advance about all the tasks that need to be scheduled. Such algorithms are less efficient as compared to static methods, but they are better suited to environments where tasks are received in a non-periodic fashion and the scheduling decision needs to be taken at run time. Hence, the effectiveness of both resource mapping techniques varies with the specific scheduling environment. Some of the dynamic scheduling algorithms include Earliest Finish Time (EFT), Particle Swarm Optimization (PSO), and Delay Optimal Task Scheduling (DOTS) [18]. 68% of the algorithms explored in this study are dynamic, while 6% and 1% are static and both static and dynamic methods respectively as shown in Figure 5. 25% of the studies have not mentioned the type of resource mapping.

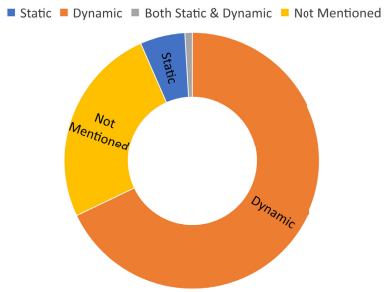

FIGURE 5. Resource mapping distribution in 106 articles included in this study.

IV. NATURE OF SCHEDULING ALGORITHMS

A task scheduling algorithm can be either preemptive or non-preemptive.

A. PREEMPTIVE SCHEDULING

In preemptive scheduling, the CPU is assigned to the incoming tasks for a limited amount of time. The scheduler has the authority to assign and withdraw the CPU during a task's lifetime. However, the context of such tasks is saved and restored during the switching process. Only 3% of the task scheduling algorithms are found to be preemptive in this study and all are used in the cloud environment as shown in Figure 6.

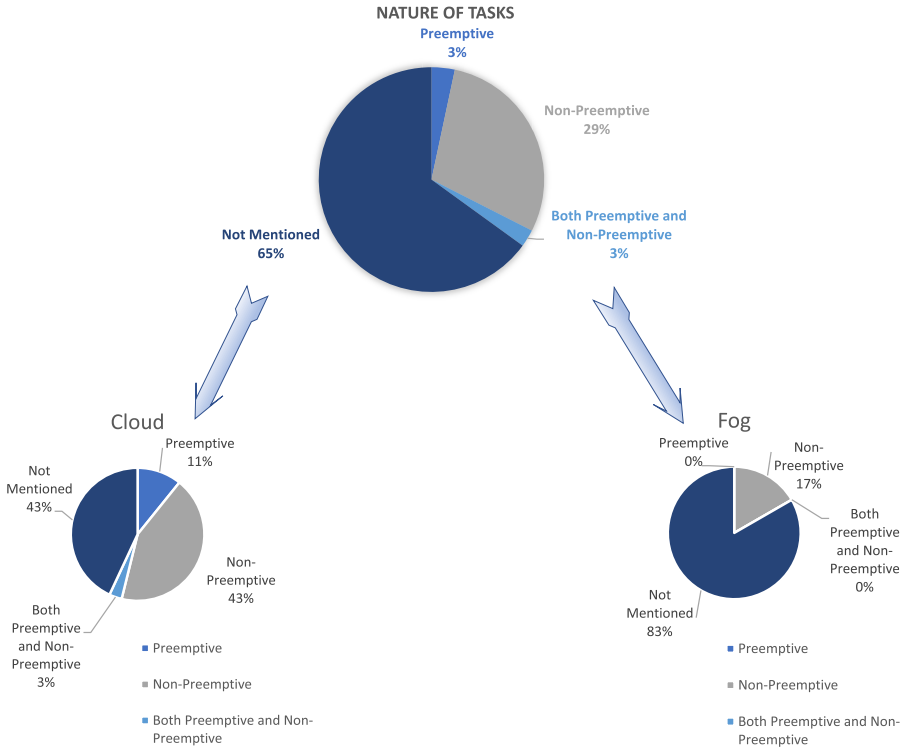

FIGURE 6. Ratio of preemptive/non-preemptive algorithms in 106 included studies.

B. NON-PREEMPTIVE SCHEDULING

A non-preemptive scheduling algorithm assigns the CPU to a task until it completes its execution while the other tasks wait in the ready queue. From Figure 6, it is evident that majority of the algorithms examined in this study are non-preemptive. It omits the overhead of saving and restoring the context of preempted tasks that is particularly useful in fog computing due to its limited resources. Moreover, there are few studies that have incorporated both preemptive and non-preemptive scheduling. There is a substantial number of studies that do not describe the nature of the algorithms, especially in fog computing relying on the assumption that the tasks are non-preemptive unless explicitly mentioned. The same assumption also applies in cloud computing but to a relatively lesser extent than in fog computing.

V. TYPES OF TASKS

The tasks received by the cloud and fog nodes can be of three types - independent, dependent (workflow), and bag of tasks. An independent task is a standalone task without any dependency on other tasks. Whereas a dependent task is part of a workflow and cannot execute before certain tasks due to its dependency on the earlier tasks. A bag of tasks on the other hand are parallel tasks with no dependency among them [19]. The majority of the studies have considered independent tasks for scheduling on the cloud and fog followed by workflows. Only three studies performed the scheduling of both independent tasks and workflows. The ratio of different types of tasks is presented in Figure 7.

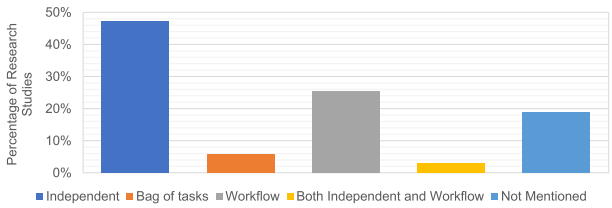

FIGURE 7. Distribution of types of tasks in 106 articles included in this study.

Table 2 in Appendix-A provides a comprehensive summary of various features of task scheduling algorithms commonly used in the cloud and fog computing in the shortlisted articles.

VI. NUMBER OF OBJECTIVES

Task scheduling is basically an optimization problem. There is an objective function (minimization or maximization) at the core of every optimization problem. A task scheduling algorithm can be intended to achieve a single objective or multiple objectives. It is observed that majority of the studies (97%) consider multiple objectives like reducing the delay and SLA violations along with meeting the task deadlines. Only four studies (3%) focused on a single objective, with one study each for optimizing the makespan [20] and reliability [21], and two studies ([22], [23]) on meeting the task deadlines. A detailed discussion on makespan, resource utilization, delay, load balancing, and energy consumption is provided in Section IX.

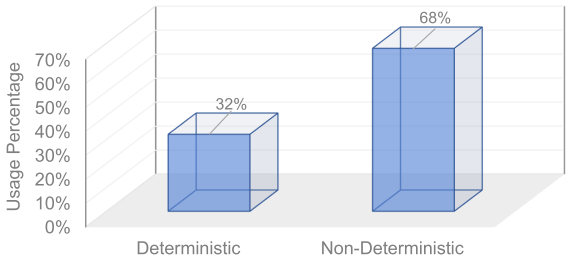

FIGURE 8. Ratio of different algorithmic approaches.

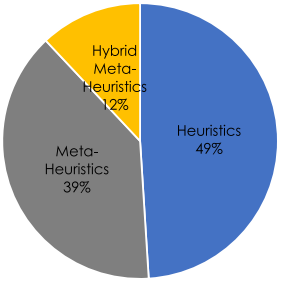

FIGURE 9. Three categories of solutions.

VII. ALGORITHMIC APPROACHES

An algorithmic approach towards task scheduling can be deterministic or non-deterministic. A deterministic routine provides the same output under different runs for the same input [24]. For task scheduling, it exploits all permutations for a typical combinatorial problem. It can solve a problem in polynomial time, but such programs work at their best when there are limited number of parameters. If the number of parameters increases above a certain threshold, a deterministic routine becomes impractical due to its increased time complexity. On the cloud, the number of resources can be large, but the incoming tasks can be huge.

Non-deterministic algorithms, on the other hand, are based on guided-random set of steps that explore and exploit the solution space to provide solutions that are sufficiently good within an acceptable amount of time. They may produce different output for the same input in different executions. The provided solutions are not the best, but good enough to serve the purpose. Such algorithms are widely used to tackle NP-hard combinatorial problems. Figure 8 shows that two third of the methods explored in this study are non-deterministic algorithms. In task scheduling, a good enough solution is preferred as opposed to a best solution with high time complexity.

VIII. CATEGORIES OF SOLUTIONS

In this review, the various task scheduling algorithms found in the shortlisted papers are categorized into three types: heuristic algorithms, meta-heuristic algorithms, and hybrid meta-heuristic algorithms. Nearly half of the solutions fall in the category of heuristics, while 39% of the studies are based on meta-heuristics. 12% are hybrid meta-heuristics as shown in Figure 9.

Heuristic algorithms are problem dependent techniques such as list based solutions, priority driven techniques or threshold based algorithms etc. These algorithms are tailored for specific problems and may not perform well for other problems. The second category (meta-heuristic algorithms) constitute high level strategies, independent of any specific problem, making them applicable to a wide range of problems. Particle Swarm Optimization and Genetic Algorithm are the two prominent meta-heuristic algorithms used across many disciplines. The third category (hybrid meta-heuristic algorithms) includes studies that employ more than one meta-heuristic algorithm to find a solution. These algorithms result from combining two or more meta-heuristic algorithms. If a particular meta-heuristic algorithm has deficiencies, such as poor convergence or limited exploration ability, hybridization could be a promising solution. By hybridization, a poor performing algorithm can offset its weakness(es).

A. HEURISTICS

The algorithms based on heuristics are problem dependent and only applicable for a specific problem. For other problem domains, they are not useful. Though heuristic algorithms provide exact solutions for a particular problem, they struggle to solve complex optimization problems. For task scheduling on the cloud and fog, heuristics appeared to be widely used with various types of algorithms as shown in Figure 9.

In heuristic based scheduling algorithms, some studies have utilized more than one technique. For example, analytical hierarchy process and divide and conquer strategies were used together along other techniques in [1]. Such studies are categorized as a sub-category ''Multi-Heuristics'' under ''Heuristics''. Further, there are 12 heuristic algorithms that appeared only once out of 106 studies. These techniques are illustrated in the ''Other Heuristics'' sub-category under ''Heuristics'' instead of creating a separate sub-category. Following are the eleven sub-categories of heuristics:

1) EARLIEST DUE DATE (EDD) BASED ALGORITHMS

The number of IoT applications have been surging progressively. Some of these applications generate delay sensitive tasks which need to be served with minimum response time. A reliability aware task scheduling algorithm based on EDD proposed for hard deadline tasks [21]. The tasks were grouped into two categories based on common deadline/execution time. The task ordering and mapping were the two proposed approaches that could be further improved through task replication and repetition. A similar study in [25] managed delay sensitive applications cost-efficiently in a mobile environment having heterogeneous fog servers. A task offloading and scheduling framework (CACOTS) reduced service bootup time with minimum cost and increased resource utilization. But makespan of the tasks was not considered. Another EDD based novel task scheduling and offloading algorithm (CEMOTS) devised to offer cost efficient services [26].

Under deadline constraints, the algorithm offloaded tasks at the right time, followed by their scheduling on the appropriate resources to reduce computational and communication costs. Nevertheless, energy consumption was not optimized.

2) EARLIEST DEADLINE FIRST (EDF) BASED ALGORITHMS

Most of the proposed solutions for deadline sensitive bag-of-tasks do not consider the dynamic aspects of clouds. A scheduling algorithm proposed in [27] maximized the number of meeting deadlines with minimal infrastructure cost in a heterogeneous environment. But response time was ignored in the comparative analysis against Bossche05 and BosscheChebyshev schedulers. A task scheduling algorithm (HCCETS) devised to minimize cost and execution of the tasks within deadlines [28]. However, under large number of patients, the deterministic routine could not handle large traffic. Similarly, RT-SANE, a task scheduling algorithm proposed to consider security and deadline constraints in [29]. Based on security and delay, RT-SANE used the resources on fog and cloud nodes accordingly. Tasks and resources were both classified into three categories based on privacy, security, and trust levels. Though the study also included the straggling tasks and their migrations, the comparative analysis was quite shallow.

3) EARLIEST FINISH TIME (EFT) BASED ALGORITHMS

The cloud service providers are concerned for the efficiency and reliability of services. RALBA, a task scheduling algorithm, balanced the workload on available computational resources on a cloud server [30]. Being a batch dynamic algorithm, it had two phases. In the first phase, bigger VMs were assigned to bigger tasks, while in the second phase, the VMs providing EFT for jobs were selected. It outperformed the traditional heuristics in terms of makespan, throughput, and resource utilization. However, it favored larger tasks and penalizing the smaller ones. In [31], a fault tolerant task allocation procedure based on EFT was proposed. It used two fault-tolerant scheduling models, considered uncertainty and an overlapping mechanism. Further, a Dynamic Fault-Tolerant Elastic algorithm was proposed for the fault-tolerant models to improve resource utilization and fault tolerance in real time task scheduling. However, the fault tolerant models did not support multiple backups. The authors in [32] proposed a dynamic task scheduling algorithm to achieve load balancing of available resources, thus increasing Average Resource Utilization (ARUR), reducing the task execution time, and a reasonable Average Response Time (ART). Larger VMs were assigned to larger tasks followed by the allocation of remaining tasks through EFT. Being a variation of EFT with 1024 tasks and 32 VMs, the algorithm behaved as a pure EFT for most of the time. In [33], an algorithm proposed for scientific workflows utilized a realistic model with random weights by extending min-min algorithm and Heterogeneous EFT (HEFT) algorithm to minimize budget and makespan.

4) ANALYTICAL HIERARCHY PROCESS (AHP) BASED ALGORITHMS

Task scheduling and resource allocation is complex in a heterogeneous environment. Analytic Hierarchy Process (AHP) used to calculate priority weights followed by ranking of computing nodes by TOPSIS (To Order the Preference by Similarity to Ideal Solution) [34]. It improved security, cost, and performance, but makespan was not discussed. Another task scheduling algorithm (PQFAHP) employed priority queue, fuzzy logic, and AHP to minimize makespan, energy utilization, and memory [35]. However, the computation capability of simulated nodes was not given. The AHP based techniques are often time consuming beyond a certain threshold, due to its pair wise comparison operation.

5) LIST BASED ALGORITHMS

While using multiple manycore processors for cloud task scheduling, many tasks run in parallel sharing the computing resources. The allocation of core(s) to different tasks for better system performance and energy consumption is a challenge. It becomes even more complex when energy and time constraints are added. For both continuous and discrete speed intervals, a list based pre-power and post-power determination technique was proposed for parallel workflows in [36]. The method of equal speed is applied to the given techniques yielding better performance. But the benchmark techniques were not mentioned in this study. Similarly, a task scheduling algorithm in [16] utilized lists and Directed Acyclic Graphs (DAG) to identify suitable nodes for task scheduling to optimize computation and finish times. The independent and dependent tasks were identified followed by task prioritization and allocation to a suitable processor. The algorithm minimized makespan and cost of computation. However, transmission cost was not discussed.

6) PRIORITY BASED ALGORITHMS

With the advancement in IoT devices, low latency applications have been developed that require real time response. A novel classification mining algorithm (I-Aprior) was proposed followed by another task scheduling algorithm (TSFC) based on I-Aprior in [37]. The task with minimum completion time ran on a node that can execute it in minimum time. The TSFC reduced task execution time and waiting time, but load balancing of multiple nodes was not addressed. Similarly, a laxity-based priority algorithm combined with Ant Colony Optimization (LBP-ACS) scheduled IoT tasks having both priority and deadline constraints [38]. Laxity-based priority addressed the task delay sensitivity, while the ACO used to minimize the energy consumption by finding a global approximate scheduling scheme. In another study, a Priority based Load Balancing (PLB) algorithm utilized multi-queues to improve makespan, response time, resource utilization, and bandwidth by resolving starvation of low priority tasks with idle resources [39]. But the computation capability of simulated VMs was not given. Two task

scheduling algorithms (PSG and PSG-M) based on greedy approach were proposed to minimize energy consumption, makespan, and deadline violation time in [40]. PSG was a priority-aware semi-greedy approach, while a combination of multi-start procedure to PSG resulted in PSG-M. However, neither PSG nor PSG-M addressed delay and latency.

7) GRADIENT BASED ALGORITHMS

For efficient resource utilization, task scheduling plays a very important role. In [20], a Gradient Based Optimization (GBO) was applied for the first time to improve makespan of tasks. Being a continuous optimization technique, GBO used rounding off technique to transform a real vector into the nearest integer value as a scheduling solution with fast convergence and avoiding local optima. For large tasks, the performance of the GBO was better in comparison with PSO, WOA, and Grey Wolf Optimization (GWO) utilizing different sizes of GoCJ workload. However, the study focused on a single objective optimization problem and evaluated against the old versions of the given meta-heuristic algorithms. Deterministic surrogate models do not cater uncertainties in QoS distribution, resulting in increasing SLA violations. The limited exploration ability of gradient driven optimization and the computationally expensive Deep Neural Network (DNN) training hinders finding the minimum response time and energy. To solve these issues, a novel task scheduling algorithm (GOSH) based on GBO utilized heteroscedastic deep surrogate models and second order derivative functions to achieve minimum scheduling time and better QoS [41]. The simulation results validated the GOSH suitability for resource contrained environments, while GOASH*, an extension of GOSH showed promising performance for a resource rich system. Both techniques achieved minimum SLA violations, energy consumption, and response time.

8) THRESHOLD BASED ALGORITHMS

The future of high performance computing is apparently an interconnected system of heterogenous nodes under the paradigm of the cloud and fog computing. An architecture proposed in [42] utilized containers to design two types of service models to reduce service delay and increase resource utilization of nodes. A real time task scheduling algorithm presented to balance the terminal devices' energy consumption with the help of a dynamic threshold policy. It also reduced the transmission delay and latency, however, the effect of energy balancing on throughput was not mentioned. To improve makespan and scheduling performance, two algorithms (TBTS and SLA-LB) were proposed based on a threshold and service level agreement respectively [43]. The TBTS was a prediction based technique that worked in two phases with threshold data based on ETC matrix, while the SLA-LB scheduled online tasks dynamically based on budget and deadline to minimize makespan, and improve resource utilization and load balancing. But the preemptive/non-preemptive nature of the algorithm was not stated. Similarly, a computing system developed utilizing the computational resources of a cloud (VMs and containers), raspberry pi's, smartphones, and grids in [44]. A bag of tasks workload was executed on the given resources to achieve minimum cost and response time. Nonetheless, the comparative analysis only included weighted round robin algorithm.

9) REINFORCEMENT LEARNING (RL) BASED ALGORITHMS

IoT devices generate enormous data that not only chokes the network, but due to a large hop count, latency also increases. An overloaded or underloaded machine is always precluded as it not only affects the processing time, but also results in a system crash. To enhance fault tolerance with minimum cost, a model proposed to incorporate predictive maintenance in a heterogenous environment by proposing two dynamic programming based problems [45]. The problems were converted into approximate dynamic programming problems using RL. It reduced the computation complexity of the problems and improved the scheduling efficiency. However, latency of the tasks was neglected. In [46], a framework proposed to enable mobile crowdsensing utilizing deep RL to manage the different resources in a network, based on bandwidth and computation cost. But energy consumption in relation to the computation cost and transmission cost was not highlighted.

A machine learning based adaptive algorithm scheduled sequential tasks and load balanced the collaborative servers, considered power usage of user device, remaining task data size, and server load [47]. The tasks were executed cooperatively by the server and user device using Software Defined Networking to minimize power usage and latency. But, the impact on response time was not examined in the context of cooperative computing. In [48], a task scheduling framework (QEEC) proposed to improve energy efficiency. A M/M/S queue model was implemented in the first phase to assign user requests for each node. In the second phase, a scheduler based on Q-learning on each node filters out the requests based on task laxity and deadline. The tasks were assigned to the nodes followed by rewarding the assignments that improved response time and resource utilization. Nonetheless, energy consumption in response to the higher resource utilization was not addressed.

In [49], tasks were scheduled using Double Deep Q-network (DDQN) in a real experiment. The adaptive learning ability of DDQN produced an optimal task schedule with minimum average response time and maximum completion rate. Despite that, the resulting degree of energy consumption was overlooked in the study. MOABCQ, a multi objective scheduling approach based on Artificial Bee Colony (ABC) and Q-learning optimized ARUR, makespan, cost, throughput, and VM imbalance [50]. But, the proposed algorithm's performance was below par in some of the tests against alternative algorithms. Similarly, IoT tasks were

divided into smaller tasks and scheduled on multiple secure computing nodes in [51]. A RL model scheduled the smallar tasks, incorporating privacy and latency constraints, and avoiding overloading the nodes to minimize delay, however, makespan and meeting deadlines were not addressed.

A container based algorithm (CBTSA) executed tasks dynamically on fog and cloud nodes for a given set of resources and deadline constraints. But, it did not cater the unbounded nature of cloud resources and computational cost of its usage. So, an improved CBTSA (ECBTSA-IRA) proposed utilizing RL and workload prioritization to address these problems by making a tradeoff among energy consumption, makespan, and cost [52]. However, the subsequent effect on response time was not elaborated. A task scheduling and preemption model (OSCAR) used clustering, heap-based optimizer, and a deep Q-network to decrease response time, makespan, waiting time, and SLA violations in [53]. It also improved system throughput while satisfying deadlines. Nonetheless, energy consumption with the improved throughput was not discussed in the study.

With a couple of exceptions, all the studies based on reinforcement learning did not validate the proposed techniques against meta-heuristics, and hybrid meta-heuristics.

10) MULTI-HEURISTICS

A multi-heuristic uses more than one heuristic technique to schedule tasks on the cloud and fog computing. Such kind of studies are presented here.

Resource allocation and task scheduling are critical to achieve better performance. A heuristic proposed in [54] combined Bandwidth-Aware Divisible Scheduling (BATS) plus BAR optimization, divide and conquer techniques, Modified AHP (MAHP), and Longest Expected Processing Time preemption (LEPT) method to improve response time and turnaround time. BATS plus BAR optimization performed the task allocation, while MAHP processed each task before its actual allocation to a resource. The resources were preempted using LEPT, while divide and conquer further improved the solutions. The proposed algorithm outperformed the prevailing BATS algorithms, but the resulting cost factors were not highlighted.

Service providers offer different resources as either on-spot, on-demand or reserved. On-spot proves to be a cheaper alternative than on-demand, but due to dynamic pricing, failures can also occur. The right mix of on-spot and on-demand resources for workflows is a challenging task. To minimize the rental cost for deadline constraint workflows, an idle-time-block based technique was proposed in [55] to develop different workflow schedules. The mix of on-spot and on-demand resource provisioning is directly proportional to resource utilization, but it was not elaborated in the study. In [11], a real time task scheduling algorithm was proposed to minimize energy consumption and execution time. A prototype scheduler application validated the effectiveness of the algorithm using various kinds of DAGs.

In [56], an interlacing peak task scheduling algorithm (MIPSM) was presented. It worked in three phases. In the first phase, the requirements and status of tasks and resources were taken periodically and updated. The second phase sorted out (classified) all the resources into different queues based on computation, input/output, and memory. In the last phase, after task scheduling, interlacing was performed with the peak resource loads. The interlacing peak algorithm improved resource utilization, load balancing, response time, and deadline violations. But, the type of the tasks was not stated. In another study, an Energy-efficient Task Scheduling Algorithm (ETSA) followed a normalization process to make scheduling decisions, taking into account task completion times and resource usage [57]. It offered an elegant trade-off between energy efficiency and makespan in a heterogeneous environment. However, the cost related to the resource usage was not included in the study.

A load balancing and cost saving task scheduling algorithm (HDCBS) in [58] improved system throughput. First, average figures were computed for task response time and wait time in queues for each resource. In the next stage, a task scheduling decision was taken based on convex optimization theory to minimize cost and maximize throughput. An onliNe multi-workflOw Scheduling Framework (NOSF) scheduled deadline sensitive workflows at run time without prior information about the task execution time and arrival time in [59]. A heuristic algorithm elastically allocated apposite resources to the workflows, improving resource utilization, deadline violation and minimizing rental cost. It outperformed IaaS-Cloud Partial Critical Paths with Deadline Distribution (IC-PCPD2) algorithm and unceRtainty-aware Online Scheduling Algorithm (ROSA). However, IC-PCPD2 being an old technique was not suitable for a comparative analysis of the proposed algorithm.

For fault tolerant scheduling, task resubmission and replication are the widely used techniques. The task replication can be in the form of a primary-backup setup where the backup instance executes if a primary fails. Both the task resubmission and replication increased resource utilization and decreased task execution time. A Hybrid Fault-Tolerant Scheduling Algorithm (HFTSA) was proposed for independent tasks with strict deadline constraints, utilizing both resubmission and replication in [60]. Online adjustment scheme and elastic resource provisioning were used by HFTSA to improve fault tolerance and resource utilization respectively. However, the effect of resubmission and replication on energy consumption was not elaborated.

11) OTHER HEURISTICS

Rejecting a lease request is always a last resort when users' demand for deadline sensitive tasks cannot be met. In [12], a backfilling task scheduling approach utilized Multiple-Criteria Decision-Making (MCDM) and Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) for deadline sensitive tasks. The study validated higher task completion rate and lower task rejection rate in comparison

to the existing backfilling algorithms. Execution time and deadlines were considered with better resource utilization. A prediction model proposed in [61] provided efficient resource management without violating SLAs. The model based on past usage, considered SLA at both scheduling and load balancing stages to minimize makespan, and load imbalance. A novel task scheduling algorithm (DOTS) minimized average task delay, transmission energy, and improved load balancing [18]. However, the proposed algorithm was not evaluated with any benchmark technique. In another study, the Lyapunov drift-plus-penalty stabilized a network and maximized completing tasks within deadlines [62]. An efficient IoT architecture, and task scheduling and allocation protocol (MobMBAR) were proposed to support patient's mobility by distributing the tasks dynamically on fog and cloud devices [63]. MobMBAR minimized makespan, energy consumption, and response time.

Game theory was employed to schedule IoT-tasks using multiple criteria in [64]. The preference functions ranked nodes based on resource utilization and latency. Using matching theory, distributed and centralized scheduling algorithms were devised, that incorporated nodes preferences, aiming to improve computation time, makespan, latency, and resource utilization. The Comparative Attributes Algorithm (CAA) was applied for task scheduling on a priority basis, while the Linear Attribute Summarized Algorithm (LASA) was used to pick the optimum nodes with maximum computation power to minimize delays in [65]. Nevertheless, the cooperation among the computing nodes was limited.

A task scheduling model and algorithm based on queuing models were formulated to estimate delay and energy consumption in [66]. Further, a tree based search proposed to improve local non-dominated solutions. But resource utilization in relation to the minimum energy consumption was not discussed. Two heuristic algorithms Min-CCV and Min-V minimized computation, transmission, and violation costs during task scheduling in volunteer computing [17]. They efficiently allocated the tasks, satisfying deadline constraints with minimum cost. In another study, a Hyper Heuristic Scheduling (HHS) algorithm minimized execution cost, energy consumption, and latency [67]. However, transmission cost was not included in the cost. A Critical Task First Scheduler (CTFS) was presented to classify critical/non-critical tasks and prioritized large sized tasks [68]. It reduced latency, network usage, and energy consumption. However, it did not provide any solution to obviate the starvation of smaller tasks. In [69], an artificial neural networks-based algorithm reduced latency through data partitioning to compute hyperparameters in parallel, thus minimizing response time and latency. However, the completion rate of incoming tasks was not optimized.

B. META-HEURISTICS

This category comprises of studies based on meta-heuristic algorithms. It showcases scheduling techniques that utilize a single meta-heuristic algorithm in addition to other strategies. There are 9 meta-heuristic algorithms that appeared only once out of 106 studies. These techniques are illustrated in the ''Other Meta-Heuristics'' sub-category under ''Meta-Heuristics''. A meta-heuristic algorithm is a set of processes that are independent of any specific problem, making it applicable to a variety of problems. These algorithms work through the phases of exploration and exploitation.

1) PARTICLE SWARM OPTIMIZATION (PSO) BASED ALGORITHMS

Dynamic task scheduling within a heterogenous environment is a challenging problem. A task scheduling algorithm was devised for manufacturing clusters in [70]. An energy consumption model was presented in addition to an improved PSO algorithm to optimally schedule tasks and balance the workload. Similarly, a binary version of PSO algorithm with low time complexity improved completion time and execution cost of tasks in [71]. In another study, a hybrid task scheduling algorithm proposed in [72] combined fuzzy theory and a modified PSO to improve throughput and load balancing. There were three phases in the proposed approach. The first phase focused on enhancing the global search through updates to the velocity methods and roulette wheel selection. The second phase improved the proposed PSO by crossover and mutation followed by a third phase to apply fuzzy logic to enhance the fitness function. This hybrid technique improved makespan, execution time, load balancing, throughput, and resource utilization. However, the dataset used in the simulation was not elaborated.

An algorithm based on Canonical PSO solved the resource allocation and management issues, and minimized makespan in an IoT based system [73]. It outperformed the traditional list based scheduling algorithms, but frequently entrapped in local optima. A scheduling model and algorithm (QoS-Discrete-PSO) was proposed to meet QoS requirements (time, cost, and reliability) and improve fault tolerance in [74]. A secure task scheduling approach, called FUPE, was designed to address malicious such as node breakage in [75]. It used fuzzy logic and Multi-Objective PSO (MOPSO) to enhance security during scheduling IoT tasks. FUPE achieved an improvement of 17% in average response time and 22% in network utilization.

In [76], two parallelized PSO algorithms minimized the time complexity of PSO. The GPU based parallel PSO outperformed many multicore PSO algorithm to minimize total cost and running time of the algorithm. The proposed G-RMPSO used fine-grained GPU threads to accelerate RMPSO particles, achieving improved parallelism. However, wait commands during synchronization decreased the efficiency of parallelized PSO algorithms. Further, the master node could become a bottleneck, especially when number of slaves are large in number. An adaptive PSO based technique, as presented in [77], aimed to minimize execution time while maximizing resource utilization and throughput. An adaptive inertia weight technique was added to the PSO

for a better tradeoff between exploration and exploitation. The performance of the algorithm was better in comparison to five renown variants of PSO except for makespan. Another hybrid approach in [14] combined Interval Type-2 Fuzzy C-Means (IT2FCM) and PSO to improve reliability and efficiency of task scheduling.

2) EVOLUTIONARY ALGORITHM (EA) BASED TECHNIQUES

A task scheduling approach based on advanced phasmatodea population evolution minimized makespan while improving resource utilization and load balancing [13]. The algorithm used convergent evolution and a restart strategy for better exploration and exploitation. The evaluation was performed on thirty benchmark functions in a heterogenous environment against Gravitational Search Algorithm (GSA), PSO, Butterfly Optimization Algorithm (BOA), and Genetic Algorithm (GA). In [78], an Opposition-Based Chemical Reaction (OBCR) method employed upward ranking, Opposition-Based learning, and Chemical Reaction optimization. The combination of these techniques resulted in diverse population avoiding local optimum solutions. OBCR performed four operations to better explore and exploit the solution space. It minimized service latency and improved the stability of dynamic nodes. However, opposition-based learning incurs high computational cost due to the evaluation of multiple solutions.

3) GENETIC ALGORITHM (GA) BASED TECHNIQUES

A decentralized task scheduling algorithm based on immune mechanism was presented in [79]. It employed both forward and backward propagation to generate optimal scheduling schemes. The proposed algorithm minimized execution time, makespan and improved load balancing. But the computation capability of simulated nodes was not provided in the study. Similarly, a Deadline and Cost aware GA (DCGA) minimized the cost of workflow execution in [80]. The tasks were classified into different levels, without exhibiting any dependencies at the same level. HEFT generated minimal cost and completion time for each task, followed by crossover and mutation operations to enhance solution diversity. However, load balancing and resource utilization were not considered. In [81], a Time-Cost aware Scheduling (TCaS) algorithm based on GA reduced operating costs and execution time.

In [82], a hybrid model, named GAECS (Genetic Algorithm and Energy-Conscious Scheduling Heuristic), prioritized tasks and assigned them to relevant resources. It minimized both makespan and energy consumption. A Hybrid Electro Search GA (HESGA) improved makespan, resource utilization, cost, and load balancing in [83]. The GA and Electro Search facilitated local and global searches respectively and outperformed ACO, GA, and Hybrid PSO-GA (HPSOGA). An Improved Elitism GA (IEGA) was devised to minimize makespan, carbon emissions, flow time, and energy consumption in [84]. In [85], another improved GA based on linear weighting was presented to minimize delay, service cost, and communication cost. By specifying the preference weights, the algorithm achieved optimal results using linear weighting. With average preference weights, the result was not satisfactory, while with the increase in single objective weight, it showed better results. In case of ignoring the preference weights altogether, optimal solutions were attained through an improved Non-Dominated Sorting GA with elitism. However, the non-dominated solutions did not increase in the latter stages of the linear weighted GA. However, the study did not consider the resource dimensions of nodes.

In [86], another task scheduling algorithm, which consider resource management, employed a modified GA. This approach produced better results than Energy-aware Task-based Virtual Machine Consolidation (ETVMC), Traveling Salesman Approach for Cloudlet Scheduling (TSACS), and ACO in terms of load balancing, makespan, resource utilization, throughput, and scheduling length. But, the datasets used in the simulation were not provided. MCGA, another GA based algorithm, optimized scheduling time, cost, resource utilization, and bandwidth in [87]. Another improved and adaptive GA addressed the issues of lengthy delays, less reliability, high energy consumption, and resources distribution in [88]. The crossover and mutation operations were divided into different intervals.

In [89], Kubernetes was employed for the containerized deployment of tasks. First, the GA was improved (IDGSA) through interval division for task scheduling. This was proceeded by adding a penalty factor, resulting in IDGSA-P, designed to foster a collaborative environment. It minimize task execution time and delays along with improved load balancing and resource utilization. However, the type of tasks was not illustrated. In another study [90], an energy efficient task scheduling algorithm (EMCS) was presented to minimize cost, energy consumption, and completion time using GA.

4) ANT COLONY OPTIMIZATION (ACO) BASED ALGORITHMS

A substantial number of applications are deadline sensitive like in healthcare and IoT etc. Therefore, efficient scheduling of tasks is crucial to improve makespan, reduce deadline violations, and energy consumption. Two algorithms based on Linear Weighted Sum and ACO were proposed to minimize energy consumption and makespan for deadline constraint tasks in a heterogenous environment [91]. Further a scaling strategy was devised to improve task scheduling and energy efficiency. The linear weighted sum can be extended to make ACO converge faster without losing solution diversity.

Another improved ACO algorithm was employed to schedule tasks in a blockchain-assisted network in [92]. The blockchain scheduling model and algorithm minimized latency, network overhead, and execution time in addition to improved privacy. Similarly, another algorithm named MOTS-ACO, utilized adaptive probability for task

scheduling to minimize turnaround time and makespan in [93]. The study also enhanced load balancing and power efficiency, achieving higher convergence speed and yielding pareto-optimal solutions.

5) WHALE OPTIMIZATION ALGORITHM (WOA) BASED TECHNIQUES

With the increasing number of tasks and resources, task scheduling is becoming more complex. The study in [94] focused on maximizing revenue within the constaints of delay-tolerant tasks. This was achieved by applying WOA to increase efficiency and profits. Nonetheless, makespan was neglected in the study. In [95], an enhanced WOA (OppoCWOA) was applied for task scheduling to optimize energy consumption and time. However, its worth noting that opposition based learning, despite its benefits, is often computationally expensive due to the evaluation of multiple solutions. Further, it is also conspicuous that chaos theory adds an overhead to the WOA. A slight increase in parameters will have a considerable effect on the algorithm's time complexity. Parallelization of algorithm can alleviate the issues of high computation complexity. In another study [96], the WOA was employed to minimize energy consumption and cost, considering task priorities.

6) SIMULATED ANNEALING (SA) BASED ALGORITHMS

For IoT driven systems, task service cost and execution within deadline are the two task scheduling challenges. A privacy-aware task scheduling algorithm based on Simulated Annealing (SA) was presented to minimize service cost and execution time utilizing goal programming approach [97]. Nevertheless, datasets used in the simulation were not mentioned. Similarly, a Multi Objective SA (MOSA) algorithm improved delay time, satisfied deadlines, and access level controls [98]. Multiple goals were achieved by finding compromised solutions, and a new goal of access level was introduced to appropriately distribute IoT tasks. However, the communication cost was ignored in the study.

7) OTHER META-HEURISTICS

There is often a conflict between user requirements and objectives of service providers, particularly for deadline-sensitive tasks. In [99], the authors proposed a task scheduling algorithm (EATSD) for deadline constraint tasks. They employed Differential Evolution (DE) and ELECTRE-III to minimize makespan and energy consumption while improving resource utilization. But, only one benchmark technique was a valid contender in the comparative analysis. In [100], Moth Flame Optimization (MFO) scheduled tasks to meet QoS requirements. It minimized task execution time and transfer time for cyber physical applications.

In [101], Chaotic Squirrel Search Algorithm (CSSA) performed the workflow scheduling where job plans were continuously generated that rendered CSSA cost-effective. For achieving global convergence, CSSA was applied to an early ecosystem produced with messy optimization. Messy local search complemented the SSA to minimize time, expenses, energy, resource consumption, and violation levels. It increased the speed and precision of convergence while retaining population diversity. But, the workload used in the simulation was not stated. In [102], NMTFOLS (a threshold-based FireFly) algorithm scheduled tasks with load balancing. It discovered the state of workload to be normal or bursty using workload predictor based on adaptive regressive holt winters algorithm. Using the given techniques with Firefly Algorithm (FA) lottery approach, appropriate tasks were assigned to the optimal VMs. Minimizing SLA violations, resource usage and improving energy efficiency were the objectives of the study.

The authors in [103] optimized the Dragonfly Algorithm (DA) to avoid pre-mature convergence through mutation (combining Mexican Hat Wavelet transform and Biography-based optimization migration process) to schedule independent tasks. The tasks were scheduled dynamically to improve response time, execution time, and SLA violations. However, a Generic GA employing the right weights could have further improved the mutation phase. A modified Henry Gas Solubility Optimization (HGSO) based on WOA and Comprehensive Opposition Based Learning (COBL) was proposed for task scheduling in [104]. The WOA and COBL improved local search and worst solutions respectively. The proposed algorithm was evaluated against WOA, HGSO, MFO, FA, PSO, and SSA using real and synthetic datasets, showing improvements in makespan and performance. But, all the compared techniques were the primitive versions.

In [105], an improved FireWorks Algorithm (FWA) was proposed, utilizing opposition based learning and Differential Evolution (DE). Opposition based learning facilitated a diversified solution set, while DE avoided local optimum solutions. The algorithm was successful at increasing resource utilization, and minimizing makespan and cost. However, according to the author, the proposed technique is resistant to task migration and did not support node failures. A strategy based on binary Jaya algorithm scheduled incoming tasks by improving energy consumption, resource utilization, load balancing, and makespan in [106]. Initially, the load was dispersed uniformly on relevant VMs followed by the best possible mapping between the tasks and VMs. NASA-iPC dataset was used on CloudSim to schedule non-preemptive and independent tasks in both heterogeneous and homogenous environments, validating both the Friedman and Holm's tests. But, the modified Jaya algorithm was compared against RR, binary PSO, and GA instead of enhanced versions of the same algorithms. An Ant Mating Optimization (AMO) in conjunction with an optimized procedure performd task scheduling by making a tradeoff between minimizing energy consumption and completion time in [107]. The datasets used in simulation was not highlighted.

C. HYBRID META-HEURISTICS

A hybrid meta-heuristic utilizes more than one meta-heuristic algorithm to schedule tasks on the cloud and fog. Such kinds of studies are grouped in this category. Hybridizing two meta-heuristic algorithms is intended to alleviate potential weaknesses in a specific meta-heuristic algorithm. However, it often increases the time complexity of the resulting hybrid algorithm. Like the heuristic based algorithms, there are also 6 hybrid meta-heuristic algorithms that appeared once in our selected literature. These techniques are listed in the ''Other Hybrid Meta-Heuristics'' sub-category under ''Hybrid Meta-Heuristics''. Following are the hybrid meta-heuristic algorithms explored in this study.

1) HYBRID VARIANTS OF PARTICLE SWARM OPTIMIZATION

Two hybrid algorithms were proposed in [108] by applying fuzzy logic and SA to PSO that produced FL-PSO and SA-PSO respectively. The proposed algorithms were further combined with dynamic dispatch queues technique to efficiently schedule tasks. The novel techniques were evaluated to be effective particularly for high dimensional problems. Makespan, cost, load balancing, waiting time, resource utilization, and queue length were optimized in the study. Still, latency was disregarded in the study. Similarly, an algorithm based on PSO improved resource utilization and makespan of non-preemptive tasks [15]. The PSO algorithm was enhanced with Hony Bee Optimization (HBO) based load balancing procedure. The proposed algorithm converged faster yielding near optimal solutions for independent task scheduling.

The improved variants of PSO and ACO were hybridized to minimize energy consumption of resource constraint nodes in a real time environment [109]. Nonetheless, the computation capability of simulated nodes was not given in the study. Similarly, a hybrid algorithm, combining PSO and SSA addressed the issues of missed deadlines and increasing number of offloaded workflows to the cloud in a multiple fog based system [110]. Further, two Markov-chain models were presented to mitigate distributed denial-of-service (DDoS) attacks by computing average bandwidth and number of VMs available for each fog network.

2) HYBRID VARIANTS OF ANT COLONY OPTIMIZATION AND GENETIC ALGORITHM

A novel model (HFSGA) and scheduling algorithm based on Flamino Search and GA improved cost, makespan, and total number of deadlines met in [113]. The cost included transmission, computation, cost associated with missing deadlines. Yet, the corresponding energy consumption was not discussed. Another study [114], formulated a multi-objective task scheduling problem based on task priority to minimize energy consumption and delay. ACO and Monarch Butterfly Optimization (MBO) were individually enhanced and subsequently hybridized to schedule independent tasks to minimize task completion rate in a real time environment. Nonetheless, the comparative analysis only included FCFS.

3) OTHER HYBRID META-HEURISTIC ALGORITHMS

The Moth Search Algorithm (MSA) was improved using Differential Evolution (DE) to minimize makespan by scheduling tasks on different VMs [115]. The study used the concept of phototaxis for exploration and Levy flights for exploitation of the search space. DE further improved the ability of local search. In another study, Dynamic Voltage and Frequency Scaling (DVFS), and a hybrid Invasive Weed Optimization and Culture Algorithm (IWO-CA) were employed to minimize energy consumption and improve resource utilization [116]. Yet, the task completion time was not discussed.

A modified Artificial Ecosystem based Optimization (AEO) scheduled IoT tasks in [117]. To improve the exploitation ability of AEO, Salp Swarm Algorithm (SSA) was applied to find an optimum solution to minimize makespan and increase throughput.

A hybrid Firebug and Tunicate Optimization (HFTO) algorithm minimized task completion time along with improving load balancing [118]. Different variants of VMs were created and the tasks were classified into different categories. It also enhanced fault tolerance, makespan, response time, and throughput. The tasks were assigned to machines based on peak load. Light weight and computation intensive tasks were allocated to the VMs with high and low CPU utilization figures respectively. The algorithm improved makespan and computational complexity even with limited resources. However, the workload used in the simulation was not mentioned.

Chimp Optimization Algorithm (ChOA) and Marine Predators Algorithm (MPA) were combined (CHMPAD) to perform task scheduling to improve makespan and throughput [119]. CHMPAD improved the exploitation ability of ChOA and escaped local optimum solutions. In [120], an infrastructure was designed for energy management followed by proposing an algorithm based on Tabu Search (TS) that was enhanced by Approximate Nearest Neighbor (ANN) and Fruit fly Optimization Algorithm (FOA). It minimized latency, response time, execution time, and allocated memory, but neglected the transmission cost.

Figure 10 presents the different categories and subcategories of heuristic, meta-heuristic, and hybrid meta-heuristic algorithms, while Figure 11 depicts the number of studies published each year, focusing on different task scheduling algorithms on the cloud and fog.

Traditional policy based algorithms, such as earliest finish time and priority driven techniques, are designed for environments where the range of workload is roughly predetermined. They better suit an environment that is not expected to face major changes in the foreseeable workload and number of users. A large part of heuristics, especially deterministic heuristics, falls into this category. However, for

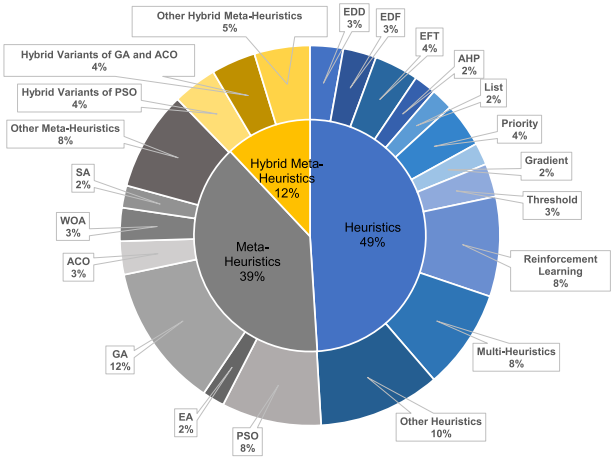

FIGURE 10. Categories and sub-categores of algorithms.

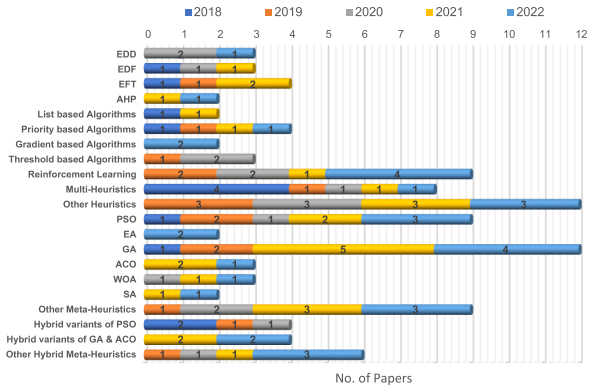

FIGURE 11. Year-wise no. of studies for each algorithm/group of algorithms.

the cloud and fog computing, the nature of workload can change at run time, unlike traditional computing systems. Machine Learning (ML) and Artificial Intelligence (AI) not only remedy the limitations of traditional task scheduling algorithms but also enhance the robustness of task scheduling. The ML models are trained with historical data, enabling them to adapt to diverse workloads in real time. On the other hand, AI possesses the ability to select strategies for computation/decision making based on prevailing circumstances. A system based on ML or AI, or both, can outperform traditional scheduling algorithms provided its computation cost is acceptable. In essence, ML and AI expedite the development of dynamic and adaptive task scheduling algorithms for the cloud and fog computing.

IX. SCHEDULING OBJECTIVES

The majority (97%) of the studies consider task scheduling as a muti-objective optimization problem. The distribution of 20 commonly used scheduling objectives in both the cloud and fog computing is presented in Figure 12.

It is observed that makespan is the prime optimization metric that received most of researchers' attention (>14%) during task scheduling on both the cloud and fog. Makespan

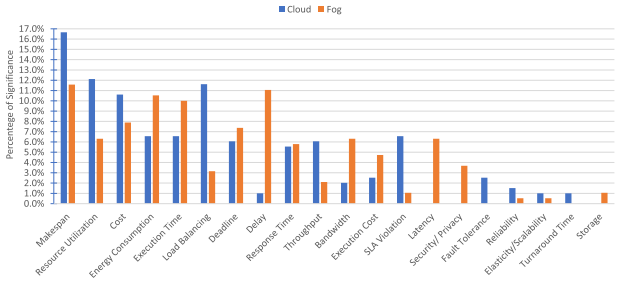

FIGURE 12. Task scheduling objectives on the cloud and fog.

refers to the completion time of all tasks submitted to a machine. It shows the efficiency of any cloud/fog node to execute a given set of tasks in the minimum possible time. Similarly, after task submission, response time and execution within deadline are equally important factors on both the cloud and fog as illustrated in Figure 12.

On the other hand, fault tolerance (2.5%) and turnaround time (1%) are addressed in a couple of studies on cloud, but are altogether neglected in the context of fog computing. Likewise, latency (6.3%), security/privacy (3.7%), and storage (1.1%) are considered in fog paradigm, but not catered in the cloud.

For task scheduling on cloud computing, makespan, resource utilization, load balancing, and cost are observed to have more than 50% of significance. For an end user, makespan and cost are crucial factors for executing workloads in minimum time, subsequently reducing cost. However, resource utilization and load balancing are decisive for cloud service providers to efficiently utilize their resources and reduce operating costs. In the cloud, poor load balancing exacerbates the overall efficiency of the infrastructure, increasing running costs.

In fog computing, makespan, delay, energy consumption, execution time, and cost are shown to have higher significance (>50%). The basic purpose of fog computing has been to serve the delay-sensitive tasks close to network edge. The fog devices being run on batteries require efficient utilization of energy to remain active for extended periods. The cost of services at cloud computing are high; therefore, fog computing strives to provide the same services at a lower cost.

Reduced makespan provides quicker results and improves overall efficiency. Better resource utilization reduces idle times, thus increasing efficiency and throughput. It also plays a substantial role in the cost-effectiveness of provided services. Load balancing avoids overloads and underloads as improper loads on machines affect stability, scalability and efficient resource usage. Energy consumption is directly proportional to environmental impacts and operational costs, posing one of the pressing issues for both the cloud and fog computing. In the cloud, it correlates with operational costs, while in fog computing, it determines the active state of fog nodes as they could have limited battery-powered energy. Cost determines the economic viability of the cloud

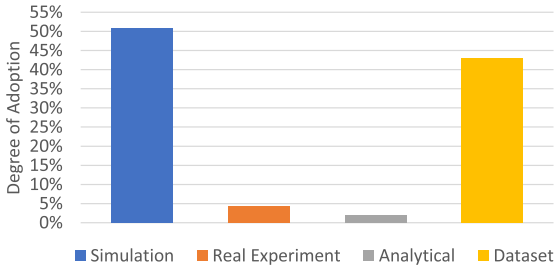

FIGURE 13. Evaluation methods.

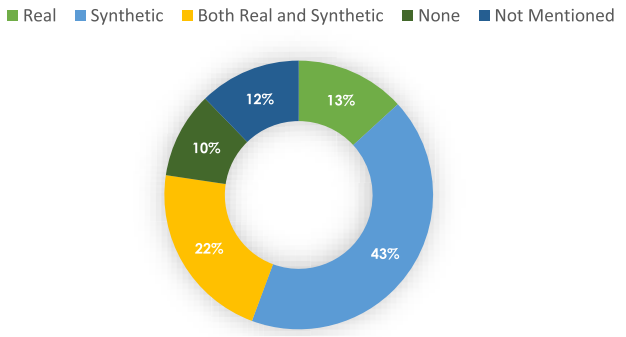

FIGURE 14. Types of workloads.

and fog architectures, which cannot be ignored. Moreover, minimizing task execution time enhances application response time, making it valuable for both the cloud and fog environments.

The relative difference among the significance of various optimization metrics is higher in cloud computing compared to fog computing. It is due to the different nature of both models. It is pertinent to consider that cloud computing is a far mature model than fog computing, which is still evolving.

Following are the seven common parameters out of twenty on the cloud and fog computing:

1) Makespan
2) Resource Utilization
3) Delay
4) Load Balancing
5) Energy Consumption
6) Cost
7) Execution Time

Besides delay which is the core reason of introducing fog computing, all the above parameters are empirical and important to evaluate any task scheduling algorithm on both the cloud and fog computing. Table 3 in Appendix-A provides various task scheduling metrics in articles reviewed for the cloud and fog computing.

Scalability and reliability are vital aspects of the cloud and fog computing; however, the shortlisted articles do not delve into them in depth. Our paper selection criteria could have overlooked studies focused on these aspects. This study specifically included articles from the past five years. Scalability and reliability might have been pressing optimization issues in the nascent stages of the cloud and fog computing. A challenging validation process for researchers could be another reason for their relatively lesser significance in this study.

X. EVALUATION TECHNIQUES

The validation and verification of the proposed models and algorithms have been performed through simulations, real experiments, analytical reasoning, and utilizing different sorts of datasets. Majority of the studies have performed simulations using real and synthetic datasets, while real experiments and analytical reasoning are conducted sparingly as shown in Figure 13.

Computer simulations are easy to conduct with a bunch of options as simulation tools, while real experiments require more resources, expertise, and time to configure a system for experiment. Thus, researchers opt for simulations to save time, resources, and cost. Real experiments are conducted in eight studies, but the relevant details are only provided by three. An experiment was conducted at a private cloud at Barcelona Supercomputing Center (BSC) in [11]. In another study, along with Python, Java, C#, and Eclipse, [28] used Arduino and DFR0027 for an experiment. Similarly, [70] employed Raspberry-Pies, and UDOO and ESP8266 boards in a real experiment to validate the proposed algorithm. Moreover, four studies [18], [36], [58], and [89] have used analytical expressions to validate various aspects of their findigs.

The datasets used in simulations and experiments can be real and/or synthetic. Real workloads are the real time logs of machines, while synthetic datasets are developed by simulations through a variety of methods. For example, a synthetic dataset can be a subset of any real dataset or can be developed based on the qualities of a real dataset. The studies included in this SLR used synthetic datasets up to 43% due to its provided ease and avoiding a compromise on the security/privacy of real data. 22% of the studies utilized both real and synthetic datasets that seems to be the safest approach for researchers. 13% of the studies have relied on real workloads only. Further, 10% of the researchers have not used any dataset, whereas 12% have not mentioned any workload in their studies as shown in Figure 14.

The respective frequencies of all workloads investigated in this study are shown in Figure 15.

Synthetic and randomly generated workloads are the choice of authors due to their provided convenience. Further, Montage, NASA Ames iPSC/860 Log, Epigenomics, and Cybershake workflows are among the widely used datasets for task scheduling on the cloud and fog. A variety of simulation tools are used for running task scheduling algorithms. CloudSim and MATLAB stands out as a dominant simulation tools in the cloud, and with the addition of iFogSim, these simulators are also widely employed in the fog, as shown in Figure 16.

A substantial number of authors also used custom-built simulations instead of the above renowned simulators. They used numerous programming languages, Integrated

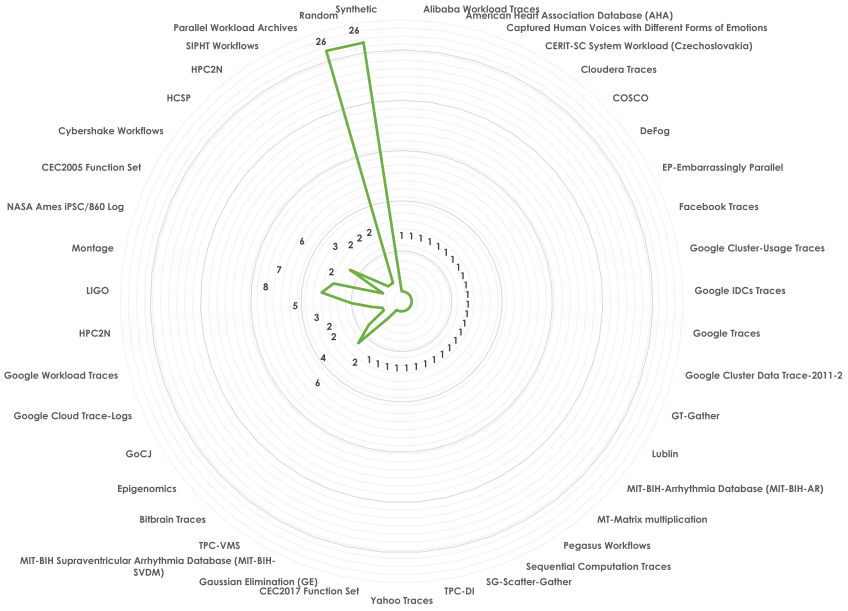

FIGURE 15. Frequencies of different real, synthetic, and random simulation workloads.

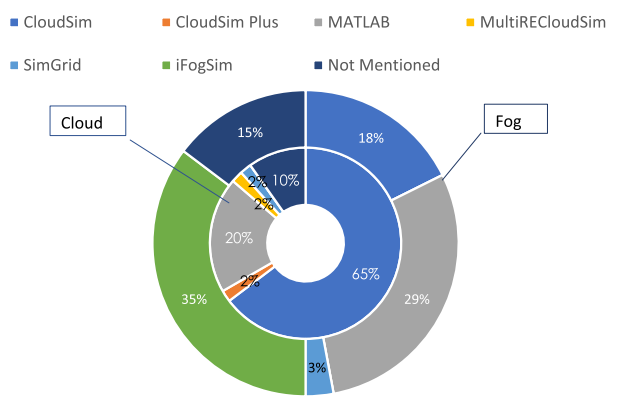

FIGURE 16. The usage of different simulation tools for task scheduling on the cloud and fog.

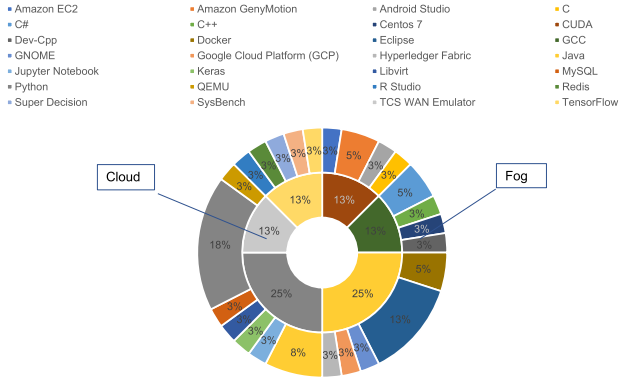

FIGURE 17. Comparison of Tools/Languages/Libraries/DBMS/IDE for custom built simulations.

Development Environments (IDE), libraries, platforms, and Database Management Systems (DBMS) as shown in Figure 17.

XI. OPEN ISSUES

Task scheduling is an optimization problem where researchers strive to improve one metric without degrading others or compromising beyond a certain threshold value. This SLR examined 20 task scheduling objectives in the selected studies. Makespan, resource utilization, delay, energy consumption, and cost are the top five objectives in task scheduling on the cloud and fog. Some of the open issues are:

1) How to improve the given QoS metrics without degrading others? is an open issue as these objectives are often inversely proportional to each other. Due to the inverse association among scheduling metrics, it is observed that one algorithm cannot minimize the makespan, cost, energy consumption, and delay along with increasing the resource utilization etc. at the same time. Therefore, multiple algorithms need to be developed to focus on a specific group of factors at one time.
2) In the context of heterogenous resources, load balancing is vital for energy savings and performance improvements. In the prevalent scheduling mechanisms targeting load balancing, mostly earlist finish time of tasks is utilized to use the provided resources in a balanced manner, but, it is at the cost of overloading fastest VMs or nodes. A mechanism is needed to balance the resource utilization of available computing resources.
3) The communication overhead between end users and the cloud and fog computing is crucial for QoS, but it is not addressed adequately by the researchers.

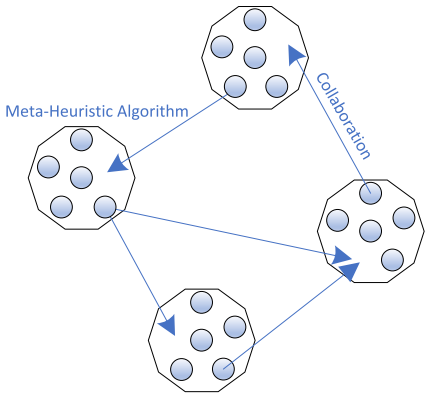

FIGURE 18. Parallel execution of meta-heuristic algorithm.

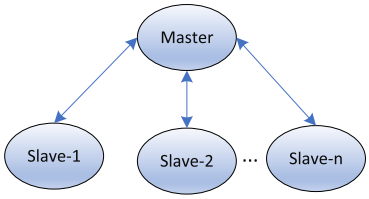

FIGURE 19. Master/slave topology.

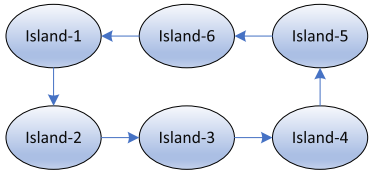

FIGURE 20. Coarse-grained topology.

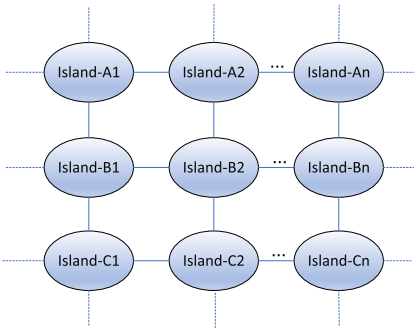

FIGURE 21. Fine-Grained Topology.

4) The issues raised due to task migration and offloading during task scheduling need further research efforts as it can improve the practicality of scheduling algorithms.
5) The carbon footprint is a serious concern as it contributes substantially to ecosystem. The usage of clean fuel by cloud data centers is an open issue to reduce CO2 emissions.
6) Task scheduling under the umbrella of minimizing SLA violations and improving fault tolerance requires the researcher's attention. Machine learning techniques can enhance the prediction and remedy of SLA violations and fault occurrence.
7) IoT applications rely on the cloud and fog resources; so further efforts are needed to make the service provisioning lightweight.
8) The issues raised by Big-Data to efficiently use the cloud computing and reduce the associated communication overhead.
9) With the emergence of 5G, the mobility of fog nodes is a hot topic of research. Unlike cloud servers, the mobility of fog nodes is linked with aggravating its security. Further, the management issues can be resolved effectively by Software Defined Networking in fog computing.
10) The factors of heterogeneity, different application requirements, virtualization technologies, and unexpected workloads are some of the open issues while scheduling tasks on the cloud and fog.
11) A scheduling algorithm is able to work across different architectures in multi-clouds, provided it accommodates the various requirements and constraints of these architectures. Some cloud providers can use virtual machines, while others could employ containers. Therefore, an algorithm should cater to these differences by adhering to open standards. Interoperability standards and cloud APIs are critical to enable seamless provisioning of resources across multiple clouds. The development of common standards facilitates task scheduling algorithms in working across different infrastructures. Although established standards are in place for multi-cloud applications, it remains a challenging aspect that requires further efforts.
12) For a hybrid fog-cloud infrastructure, the task scheduling algorithm can perform the mapping of tasks based on the specific characteristics and constraints of fog and cloud nodes to optimize the overall system performance. A prevalent solution is to direct compute-intensive tasks to the cloud and execute delay-sensitive tasks at the fog nodes. Dynamic load balancing in fog-cloud can enhance system performance. However, accomplishing it with resource-constrained devices is a challenge.

XII. CHALLENGES AND FUTURE DIRECTIONS

Numerous techniques have been investigated in the three categories of heuristics, meta-heuristics, and hybrid meta-heuristic algorithms for task scheduling on the cloud and fog. However, based on individual technique(s), multi-heuristic and meta-heuristic algorithms especially PSO and GA have been the choice of researchers. In multi-heuristics, there is a wide range of different algorithms utilized for task scheduling, but none of the given techniques got exclusively attention. Based on the PSO and GA based studies, it is examined that the computational complexity and the right mix of exploration and exploitation determines the viability

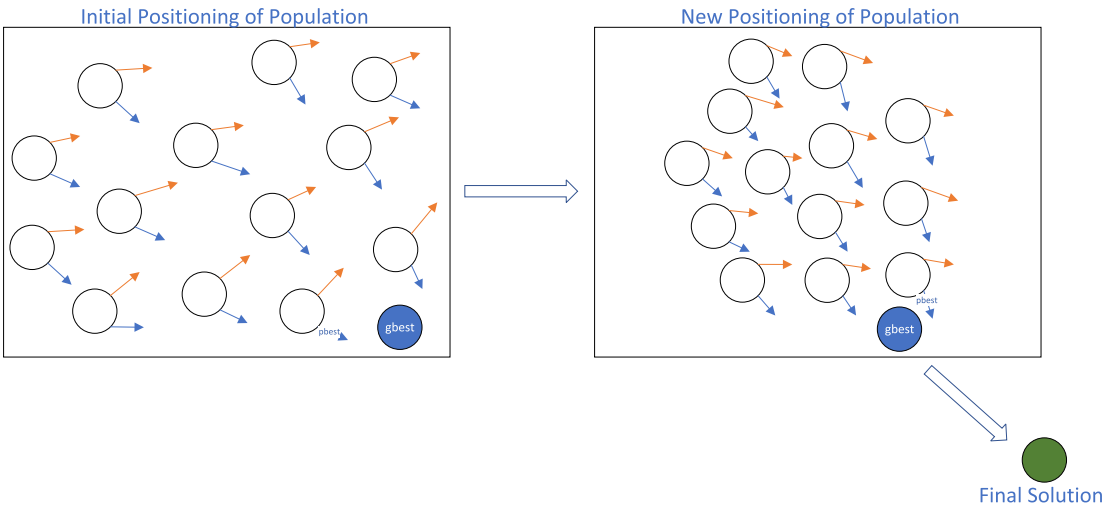

FIGURE 22. Position of particles in different iterations in PSO.

of a meta-heuristic algorithm. Following can be some of the challenging endeavours to improve the existing meta-heuristic algorithms:

A. PARALLELIZATION

The inherent complexity of a meta-heuristic algorithm increases its execution time. A meta-heuristic algorithm converges to a final solution in each iteration starting with exploration and ending in exploitation in the later stages. This process takes time and one of the methods for speedup is parallel execution as shown in Figure 18. So, an algorithm can be run on multiple machines/threads to get the solution in minimum time. Due to large search space with CPU and memory intensive objective functions, concurrent execution of a meta-heuristic algorithm is viable. Yet, if there is only one machine available, the parallelized algorithm will still perform better and provide a robust solution assisted by cooperation. Moreover, a parallel meta-heuristic algorithm is useful when the scale of a problem increases up to a point that cannot be processed by a single machine. In this case, the solution space or the required computations can be distributed on multiple machines. For example, a Parallel Squirrel Search algorithm using fuzzy logic optimally scheduled tasks in high load conditions in [121].

The following three models define the granularity of a parallelized meta-heuristic algorithm:

1) Algorithmic level parallelism
2) Iteration level parallelism and
3) Solution level parallelism

At the algorithmic level, multiple instances of a meta-heuristic can run either in an independent or cooperative manner to construct final solution. At the iteration level, a single step of the algorithm is parallelized for time efficient execution that is based on the distribution of the solution. On the other hand, solution level parallelism deals with the parallelization of the processing for a single solution. Both

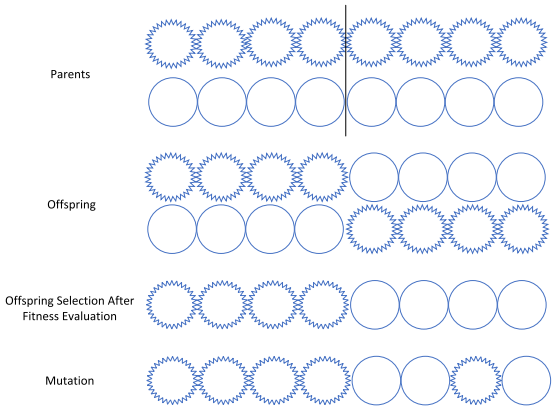

FIGURE 23. Genetic algorithm operations.

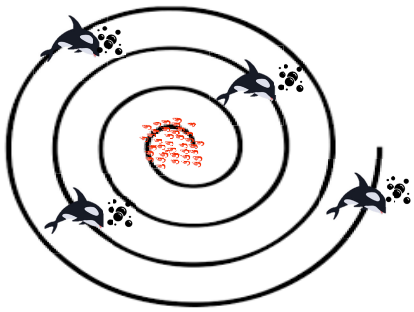

FIGURE 24. Whale optimization algorithm.

the algorithmic and iteration level parallelism are problem independent, while the solution level parallelism is problem dependent. Further, algorithmic and solution level parallelism alter the behavior of a meta-heuristic, while the iteration level parallelism does not.

There are also two parallelization strategies used with population based meta-heuristic algorithms.

TABLE 2. Comparison of tasks, types of solutions, algorithms, and evaluation methods.

References	Nature of Scheduling Algorithm		Resource Mapping		Type of Tasks			Algorithm(s)	Proposed Solution			VM/Node Characteristics	Time Complexity	Evaluation Methods			
	Preemptive	Non-Preemptive	Static	Dynamic	Independent	Bag of tasks	Workflow		Heuristic	Meta-Heuristic	Hybrid Meta-Heuristic			Simulation	Analytical	Dataset	Real Experiment
[11]		✓		✓			✓	MHRA	✓			✓		✓		✓	✓
[12]		✓	✓		✓			Improved Backfilling	✓					✓		✓	
[13]		✓	✓		✓			APPE		✓		✓		✓		✓	
[14]	-	-	✓	✓			✓	IT2FCM+PSO		✓				✓		✓	
[15]		✓	✓		✓			PSO+HBA			✓	✓		✓		✓	
[16]		✓	✓				✓	List-based Algorithm	✓			✓	✓	✓		✓	
[17]	-	-	✓			✓		Min-CCV and Min-V	✓			✓	✓	✓		✓	
[32], [31], [45], [40]		✓		✓	✓			DRALBA, DEFT, RL, (PSG and PSG-M)	✓			✓	✓	✓		✓	
[30], [49]		✓		✓	✓			RALBA, DDQN-TS	✓			✓		✓		✓	
[100]	-	-		✓	✓			MFO		✓		✓		✓		-	
[112], [116]	-	-		✓			✓	GA+ACO, DVFS+IWO-CA			✓	✓		✓		✓	
[102]	✓	✓		✓				NMTFOLS		✓				✓		✓	
[101], [75]	-	-		✓			✓	CSSA, Fuzzy logic + MPSO		✓		✓		✓		-	
[103], [87], [74]	-	-		✓	✓			BMDDSF, Enhanced GA, QoS-Discrete-PSO		✓				✓		✓	
[77]	-	-		✓	✓			AdPSO		✓		✓		✓		✓	
[111]	-	-		✓	✓			Alts			✓	✓		✓		✓	
[106], [104], [99]		✓		✓	✓			Modified Jaya, HGSO, DE+ ELECTRE III		✓		✓		✓		✓	
[60]		✓		✓	✓			HFTSA	✓			✓		✓		✓	✓
[50], [48], [43], [65]	-	-		✓	✓			ABC+Q-learning, Q-Learning, TBTS and SLA-LB, CAA+LASA	✓			✓		✓		✓	
[20]	-	-	-	-	✓			GBO	✓			✓	✓	✓		✓	
[108]		✓		✓	✓			Fuzzy-PSO and SA-PSO			✓	✓		✓		✓	
[57], [37]		✓		✓	✓			ETSA, NN	✓				✓	✓		✓	
[33]		✓	-	-			✓	Enhanced Min-Min and HEFT	✓			✓		✓		✓	
[91], [105]		✓		✓	✓			LWS and ACO, Modified FWO		✓		✓	✓	✓		✓	
[71]	-	-	-	-	✓			BPSO		✓		✓	✓	✓		✓	
[59]		✓		✓			✓	NOSF	✓			✓	✓	✓		✓	
[27]		✓		✓		✓		Online Scheduler	✓			✓	✓	✓		✓	
[21]	✓	✓		✓		✓		Reliability Aware Scheduling	✓				✓	✓		✓	
[115], [113]	-	-	-	-	✓			MSA+DE, HFSGA			✓	✓	✓	✓		✓	
[96], [83], [92]	-	-		✓	-	-	-	WOA, HESGA, Improved ACO+Blockchain		✓		✓		✓		✓	

TABLE 2. *(Continued.)* **Comparison of tasks, types of solutions, algorithms, and evaluation methods.**

[94]	-	-	-	-	✓			WOA		✓				✓		✓	
[36]		✓	-	-	✓		✓	List Scheduling	✓					✓	✓	✓	
[82]	-	-		✓			✓	GAECS		✓		✓	✓	✓		✓	
[86]		✓		✓	✓			G-MOTSA		✓				✓		✓	
[93], [95]	-	-		✓	✓			MOTS-ACO, OppoCWOA		✓		✓		✓		✓	
[55]	✓	✓		✓			✓	ITB	✓			✓		✓		✓	
[118]	✓			✓			✓	HFTO			✓	✓		✓		-	
[72]		✓	-	-	✓			Fuzzy PSO		✓		✓		✓		-	
[54]	✓			✓	✓		✓	AHP	✓			✓		✓		✓	
[73]	✓			✓	✓			CPSO		✓			✓	✓			
[39], [26]	-	-		✓			✓	PLB, CEMOTS	✓				✓	✓		✓	
[107], [62]	-	-	-	-	✓			AMO+Optimized Task Distribution, Lyapunov drift-plus-penalty	✓			✓		✓		✓	
[58]	-	-		✓			✓	HDCBS	✓			✓		✓	✓	✓	
[56]	✓			✓	-	-	-	MIPSM	✓			✓	✓	✓		✓	
[80]	-	-		✓			✓	DCGA		✓			✓	✓		✓	
[61]	-	-		✓	✓			SLA-Aware Scheduling	✓					✓		✓	
[76]	-	-	-	-	✓			M-RMPSO+G-RMPSO		✓		✓	✓	✓		-	
[88]	-	-		✓			✓	Improved GA		✓				✓		✓	
[78]	-	-		✓			✓	Opposition based Hybrid Evolutionary Approach		✓		✓		✓		✓	
[117]		✓	-	-	✓			AEO+SSA	✓			✓		✓		✓	
[66]	-	-	-	-	-	-	-	ICGM+TIGH	✓				✓				✓
[84]	-	-	-	-	-	-	-	IEGA		✓		✓				✓	✓
[98]	-	-	-	-	✓			MOSA		✓		✓		✓		✓	
[97]	-	-	-	-	-	-	-	MOSA		✓		✓		✓			
[47]	-	-		✓	✓			DRL	✓					✓			
[119]	-	-	-	-	-	-	-	CHMPAD			✓	✓	✓	✓		✓	
[67]	-	-	-	-			✓	HHS	✓			✓	✓	✓		✓	
[18]	-	-		✓	-	-	-	DOTS	✓			✓		✓	✓		
[68]	-	-		✓			✓	CTFS	✓			✓		✓			
[51]	-	-		✓	-	-	-	Q-Learning	✓					✓			
[41]	-	-		✓	✓			GBO+GOSH	✓					✓		✓	✓
[79]	-	-		✓	-	-	-	Immune GA		✓				✓		✓	
[63]	-	-		✓	-	-	-	MobMBAR	✓			✓		✓		✓	✓
[29]		✓	-	-	✓			RT-SANE	✓			✓	✓	✓		✓	
[120]	-	-		✓	-	-	-	TS+(ANN)+FOA			✓	✓	✓	✓			
[38]	-	-		✓			✓	Laxity+ACO	✓			✓		✓		✓	
[52]	-	-	✓				✓	ECBTSA-IRA	✓			✓		✓		✓	
[25]		✓		✓	✓			MSCFS and CA-COTS	✓			✓	✓	✓		✓	✓
[53]		✓		✓			✓	Modified Heap-based Optimizer+Deep Q-network	✓			✓		✓		✓	
[34]	-	-	-	-	-	-	-	AHP+TOPSIS	✓			✓		✓			
[89]	-	-	-	-			✓	IDGSA-P		✓		✓		✓	✓	✓	
[44]	-	-		✓		✓		Thresholds based scheduling	✓					✓		✓	
[46]	-	-		✓		✓		Q-Network+Experience Relay	✓				✓	✓		✓	
[64]	-	-		✓	-	-	-	Centralized+Distributed Matching	✓			✓		✓		✓	
[69]	-	-		✓	-	-	-	ANN	✓					✓		✓	

TABLE 2. *(Continued.)* **Comparison of tasks, types of solutions, algorithms, and evaluation methods.**

[35]	-	-		✓	-	-	-	PQFAHP	✓				✓	✓		✓	
[114]	-	-	-	-	✓			ACO+MBO			✓	✓	✓	✓		-	
[28]	-	-		✓	✓			HCCETS	✓			✓	✓	✓		✓	✓
[90]	-	-	-	-			✓	Improved GA		✓		✓	✓	✓		✓	
[81]	-	-	-	-	✓	✓		TCaS		✓		✓		✓		✓	
[109]		✓	-	-	✓		✓	IPSO+IACO			✓			✓			
[110]	-	-	-	-			✓	PSO+SSA+MDP			✓			✓		✓	
[42]	-	-		✓	-	-	-	Dynamic Threshold Strategy	✓			✓		✓		-	
[70]		✓		✓	-	-	-	Improved PSO		✓		✓					✓
[85]	-	-	-	-			✓	LWGA		✓		✓		✓		✓	

1) Computation based parallelism: The operations that are performed on each particle/individual of the whole population are performed in parallel.
2) Population based parallelism: The population is divided into multiple parts, each of which is handled separately. These parts exchange information and then combine to form the final solution.

Similarly, there are two modes for the function evaluation in a parallel metaheuristic algorithm: synchronous and asynchronous. If the evaluation is performed for each particle in parallel and the state is synchronized at each iteration, is referred to as synchronous parallel mode. However, synchronous parallelism is less productive. For example, a parallelized variants of GA and PSO dynamically offloaded tasks in [122]. But they were only appropriate for low rate of task generation by the IoT devices due to the reduced productivity by synchronous parallelism. In contrast, in asynchronous mode, the evaluation of populations does not synchronize with each other in each iteration, instead, it depends upon the state and update timings of separate populations/agents. Here every subpopulation/agent exhibits different behavior using same/different function evaluation. The different topologies for parallel meta-heuristic algorithms are:

1) Master-Slave: There is a single master node with multiple slave nodes coordinating with the master as shown in Figure 19. However, if the slave nodes increase over a certain limit, then communication overhead becomes a bottleneck.
   In [123], a multi-objective parallelized PSO algorithm combined with pareto optimal theory to schedule micro-services based on containers. But, the inter-process communication among swarms of particles was based on a master-slave model that resulted in small improvements with increasing number of swarms and iterations. The best possible results were achieved with pnum = 300, iter-num = 300, c1 = 0.1, c2 = 0.45, and w = 0.45.
2) Coarse-Grained (Island or Ring topology): A program is split into multiple chunks where separate computation(s) takes place. It provides low synchronization and communication overheads due to a ring structure as expressed in Figure 20. But, due to the uncertain degree of workload on different processors, load balancing could be a challenge.
   A parallel GA for job shop scheduling used two level parallelization in [124]. Executing the exigent tasks with the fastest machines impacted the overall efficiency of the scheduling. Using a hybrid GPU-CPU approach, and island topology with classic GA and cellular GA made the algorithm complex. Further, the classic GA can also be partially parallelized on multi-core CPU. But, it can trap in local optimum due the naive nature of roulette wheel technique when dealt with classic GA on island B. Moreover, at the migration, the individuals running on GPU were migrated to the CPU that degraded the performance.
3) Fine-Grained topology (Cellular topology) distributes a program into smaller tasks evenly to run on more processors achieving better load balancing. In Figure 21, it is evident that every island is connected to two to four islands that results into higher synchronization and communication overheads.
4) The combination of any of the above results into a hybrid topology.

In essence, the parallelization of a meta-heuristic algorithm addressing the above limitations is one of the future directions of this study for scheduling of independent tasks in cloud computing.

B. SETTING/CONFIGURATION OF WEIGHTS

A meta-heuristic algorithm uses random agents to converge to a global minimum/maximum. A large inertia weight enables global search, while a smaller value accelerates the local search. Similarly, a large mutation ratio enables global search, while a large crossover ratio facilitates local search. The convergence can be improved by appropriate values of the different weights to balance the exploration and exploitation without losing diversity. In Figure 22, an improved PSO algorithm utilized adaptive random weights in the later stages to obviate local optimum solutions by restricting the inertia weight between 0.4 and 0.7 [125].

An enhanced PSO (SADCPSO) combined three techniques of adaptive inertia weight, and the operators for chaos

TABLE 3. Various task scheduling metrics in articles reviewed for the cloud and fog computing.

Reference	Execution Time	Makespan	Execution Cost	Cost	Latency	Delay	Response Time	Resource Utilization	Energy Consumption	Throughput	SLA Violation	Deadline	Turnaround Time	Load Balancing	Security/ Privacy	Fault Tolerance	Storage	Bandwidth	Reliability	Elasticity/Scalability
[12]	✓							✓				✓								
[13]		✓		✓										✓						
[14]				✓						✓										
[15]		✓						✓						✓						
[16]	✓	✓	✓																	
[17]		✓		✓		✓												✓		
[18]						✓			✓					✓						
[32]	✓	✓					✓	✓	✓	✓	✓			✓						
[31]								✓								✓			✓	
[45]				✓												✓				
[40]		✓							✓			✓								
[30], [77]		✓						✓		✓				✓						
[49]		✓					✓													
[100], [37]	✓	✓																		
[112]	✓			✓			✓							✓						
[116]								✓	✓											
[102]								✓	✓		✓			✓						
[101]		✓	✓	✓			✓		✓		✓									
[75]							✓	✓							✓					
[103]	✓						✓				✓									
[87]	✓			✓				✓										✓		
[74]	✓	✓		✓												✓			✓	
[111]		✓						✓			✓			✓						
[106]		✓						✓	✓					✓						
[104]		✓	✓																	
[99]		✓						✓	✓		✓	✓		✓						
[60]	✓							✓				✓				✓				✓
[50], [86]		✓		✓				✓		✓				✓						
[48]							✓	✓	✓			✓								
[43]		✓		✓				✓			✓	✓		✓						
[65]	✓					✓														
[20]		✓																		
[108], [83]		✓		✓				✓						✓						
[57], [82], [107]		✓							✓											
[33]		✓		✓																
[91]	✓	✓							✓			✓								
[105]		✓		✓				✓		✓										
[71]	✓	✓	✓					✓						✓						
[59]				✓				✓			✓	✓								
[27]				✓		✓					✓	✓								✓
[21]												✓							✓	
[115]		✓								✓				✓						
[113]		✓		✓								✓								
[96], [11]		✓		✓					✓											
[92]	✓				✓										✓			✓		

TABLE 3. *(Continued.)* **Various task scheduling metrics in articles reviewed for the cloud and fog computing.**

[94]	✓		✓	✓		✓														
[36]	✓								✓	✓										
[93]		✓							✓				✓	✓						
[95]	✓								✓											
[55]				✓							✓	✓								
[118]	✓	✓					✓			✓				✓		✓				
[72]	✓	✓						✓		✓				✓						
[54]							✓						✓					✓		
[73]		✓						✓		✓										
[39]		✓					✓	✓										✓		
[26]			✓	✓		✓						✓						✓		
[62]	✓											✓								
[58]				✓			✓			✓				✓						
[56]							✓	✓			✓	✓		✓						
[80]		✓	✓								✓	✓								
[61]		✓									✓			✓				✓		
[76]				✓																
[88]	✓					✓		✓	✓											
[78]	✓				✓															
[117], [119]		✓								✓										
[66]						✓			✓											
[84]		✓				✓			✓											
[98], [97]	✓		✓			✓						✓								
[47]					✓				✓					✓						
[67]			✓		✓				✓											
[68]					✓	✓	✓		✓									✓		
[51]					✓	✓	✓							✓	✓					
[41]							✓	✓	✓		✓									
[79]	✓	✓												✓						
[63]		✓					✓		✓											
[29]						✓	✓					✓			✓					
[120]	✓			✓	✓		✓													
[38]						✓			✓			✓								
[52]		✓		✓					✓			✓								
[25]			✓	✓		✓		✓				✓								
[53]	✓	✓				✓	✓	✓		✓	✓	✓						✓		
[34]	✓			✓	✓										✓		✓	✓		
[89]	✓					✓		✓						✓						
[44]				✓			✓													
[46]			✓	✓	✓			✓										✓		
[64]	✓	✓			✓			✓												✓
[35]		✓				✓	✓		✓			✓					✓			
[114]		✓			✓	✓			✓											
[28]			✓	✓								✓						✓		
[90]	✓	✓		✓					✓									✓		
[81]	✓	✓	✓	✓														✓		
[69]					✓		✓													
[109]		✓				✓			✓										✓	
[110]		✓						✓				✓			✓			✓		
[42]				✓		✓		✓							✓					
[70]									✓					✓						
[85]				✓		✓												✓		

and disruption theories [126]. The three techniques facilitated adjustments in convergence, retaining population diversity, and global optimum solutions to minimize execution time. The inertia weight was maximized to favor exploration after getting particles with more fitness. The disruption operator maintained the Euclidean distance among the random particles based on a threshold, the current, and maximum iteration values to avoid local optimum solutions.

A chaotic variable also improved the randomness of the particles with both upper and lower limits. However, how to specify the threshold, upper, and lower limits need further investigation.

In another study, a task scheduler utilized PSO with several strategies (linear, sigmoid, chaotic, SA, and logarithm) for controlling inertia weights in [127]. But, the increasing number of parameters made the algorithm complex that resulted in performance degradation. In [128], another improved PSO algorithm was introduced, incorporated a cosine inertial weight strategy along with uniform initial parameter values and ranking. This appraoch aimed to optimize both the exploration and exploitation. Uniform random values averted the particles to be in clumps (results into local optimum solutions). However, the time complexity of the PSO increased significantly due to these three techniques.

The GA is inspired from the concept of evolution where parents produce an offspring having the characteristic of both parents as shown in Figure 23. Crossover and mutation control the ratio of exploration and exploitation and after hundreds of generations, an optimal solution is produced. A biased random key GA introduced with a new feature of shaking to solve the permutation for a flow shop scheduling in [129]. The shaking facilitated the solution to escape from local optima. Though the shaking was not productive at times, it can also reduce the fitness of the elite set. Further, if the shaking did not help, then a full reset was the only remaining choice. In [130], the GA combined with biased random algorithm to discover VMs with minimum load to achieve a better fitness value. It helped load balancing of resources and minimum migrations based on the average load of VMs instead of instances. However, the best candidates and the non-best candidates participated in crossover and mutation without elitism. Also, a replication based crossover was used since the normal crossover did not work after using the biased random algorithm.

Unlike PSO and GA, WOA is a relatively new meta-heuritic algorithm in which whales use maneuvers such as upward spirals and double loops as part of a bubble net strategy to capture prey as shown in Figure 24. Here, the distance control parameter predominantly determines the balance between exploration and exploitation that requires appropriate manipulation. In [131], an enhanced WOA was proposed, incorporating random and adaptive weighting strategies to avoid local optima. Subsequently, it was combined with the Bees Algorithm. The weights (w1 and w2) did not reduce linearly due to the cauchy random numbers, resulting in reduced performance. The Bees algorithm improved the final results, but at the cost of expensive mutation operation.

Several studies have been conducted to set/configure the different weights/parameters of meta-heuristic algorithms, but they have their limitations. A robust algorithm by proper configuring/optimizing the different weights of a meta-heuristic algorithm is another future direction of this study for scheduling of independent tasks in fog computing.

XIII. CONCLUSION

This study presented a comprehensive review of state-of-the-art task scheduling algorithms in the cloud and fog computing from 2018-2022. It provided the taxonomy and classification of different task scheduling algorithms. The existing studies are categorized into heuristics, meta-heuristics, and hybrid meta-heuristic algorithms with relevant subcategories. All the explored methods are analyzed based on the resource mapping and nature of algorithms. The types of tasks, number of objectives, and algorithmic approaches are examined followed by the critical evaluation of all studies. Most of the task scheduling algorithms are dynamic and non-preemptive in nature, while the higher ratio of tasks is independent as compared to bag of tasks and workflows. Moreover, a total of twenty task scheduling objectives are surveyed from 106 studies with a comparative analysis of significance on the cloud and fog computing. Makespan, resource utilization, delay, load balancing, and energy consumption are observed to be the imperative objectives on the cloud and fog. The evaluation methods used by the researchers are investigated including simulations, real experiments, analytical equations, and datasets. Synthetic and random datasets are largely employed due to their provided convenience. Further, the widely used simulation tools and other programming languages/IDEs/libraries/platforms for custom built simulations are highlighted. CloudSim, iFogSim, and MATLAB are found to be the widely used simulation tools. At the end, the open issues, challenges, and future directions are discussed.

APPENDIX

See Tables 2 and 3.

REFERENCES

[1] R. Garg, "MCDM-based parametric selection of cloud deployment models for an academic organization," *IEEE Trans. Cloud Comput.*, vol. 10, no. 2, pp. 863–871, Apr. 2022.

[2] M. R. Alizadeh, V. Khajehvand, A. M. Rahmani, and E. Akbari, "Task scheduling approaches in fog computing: A systematic review," *Int. J. Commun. Syst.*, vol. 33, no. 16, Nov. 2020, Art. no. e4583.

[3] A. Amini Motlagh, A. Movaghar, and A. M. Rahmani, "Task scheduling mechanisms in cloud computing: A systematic review," *Int. J. Commun. Syst.*, vol. 33, no. 6, Apr. 2020, Art. no. e4302.

[4] A. Arunarani, D. Manjula, and V. Sugumaran, "Task scheduling techniques in cloud computing: A literature survey," *Future Gener. Comput. Syst.*, vol. 91, pp. 407–415, Feb. 2019.

[5] N. Kaur, A. Kumar, and R. Kumar, "A systematic review on task scheduling in fog computing: Taxonomy, tools, challenges, and future directions," *Concurrency Comput., Pract. Exper.*, vol. 33, no. 21, Nov. 2021, Art. no. e6432.

[6] M. Hosseinzadeh, M. Y. Ghafour, H. K. Hama, B. Vo, and A. Khoshnevis, "Multi-objective task and workflow scheduling approaches in cloud computing: A comprehensive review," *J. Grid Comput.*, vol. 18, no. 3, pp. 327–356, Sep. 2020.

[7] 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 Evol. Comput.*, vol. 62, Apr. 2021, Art. no. 100841.

[8] X. Yang and N. Rahmani, "Task scheduling mechanisms in fog computing: Review, trends, and perspectives," *Kybernetes*, vol. 50, no. 1, pp. 22–38, Mar. 2021.
[9] B. Jamil, H. Ijaz, M. Shojafar, K. Munir, and R. Buyya, "Resource allocation and task scheduling in fog computing and Internet of everything environments: A taxonomy, review, and future directions," *ACM Comput. Surveys*, vol. 54, no. 11s, pp. 1–38, Sep. 2022.
[10] N. Almurisi and S. Tadisetty, "Cloud-based virtualization environment for IoT-based WSN: Solutions, approaches and challenges," *J. Ambient Intell. Humanized Comput.*, vol. 13, no. 10, pp. 4681–4703, Oct. 2022.
[11] F. Juarez, J. Ejarque, and R. M. Badia, "Dynamic energy-aware scheduling for parallel task-based application in cloud computing," *Future Gener. Comput. Syst.*, vol. 78, pp. 257–271, Jan. 2018.
[12] S. C. Nayak and C. Tripathy, "An improved task scheduling mechanism using multi-criteria decision making in cloud computing," *Int. J. Inf. Technol. Web Eng.*, vol. 14, no. 2, pp. 92–117, Apr. 2019.
[13] A.-N. Zhang, S.-C. Chu, P.-C. Song, H. Wang, and J.-S. Pan, "Task scheduling in cloud computing environment using advanced phasmatodea population evolution algorithms," *Electronics*, vol. 11, no. 9, p. 1451, Apr. 2022.
[14] H. Kchaou, Z. Kechaou, and A. M. Alimi, "A PSO task scheduling and IT2FCM fuzzy data placement strategy for scientific cloud workflows," *J. Comput. Sci.*, vol. 64, Oct. 2022, Art. no. 101840.
[15] F. Ebadifard and S. M. Babamir, "A PSO-based task scheduling algorithm improved using a load-balancing technique for the cloud computing environment," *Concurrency Comput., Pract. Exper.*, vol. 30, no. 12, Jun. 2018, Art. no. e4368.
[16] R. Madhura, B. L. Elizabeth, and V. R. Uthariaraj, "An improved list-based task scheduling algorithm for fog computing environment," *Computing*, vol. 103, no. 7, pp. 1353–1389, Jul. 2021.
[17] 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 Trans. Internet Technol.*, vol. 21, no. 4, pp. 1–21, Nov. 2021.
[18] 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 Things J.*, vol. 6, no. 2, pp. 3533–3544, Apr. 2019.
[19] A. S. Abohamama, A. El-Ghamry, and E. Hamouda, "Real-time task scheduling algorithm for IoT-based applications in the cloud–fog environment," *J. Netw. Syst. Manage.*, vol. 30, no. 4, pp. 1–35, Oct. 2022.
[20] X. Huang, Y. Lin, Z. Zhang, X. Guo, and S. Su, "A gradient-based optimization approach for task scheduling problem in cloud computing," *Cluster Comput.*, vol. 25, no. 5, pp. 3481–3497, Oct. 2022.
[21] C. K. Swain, N. Saini, and A. Sahu, "Reliability aware scheduling of bag of real time tasks in cloud environment," *Computing*, vol. 102, no. 2, pp. 451–475, Feb. 2020.
[22] S. C. Nayak, S. Parida, C. Tripathy, and P. K. Pattnaik, "An enhanced deadline constraint based task scheduling mechanism for cloud environment," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, no. 2, pp. 282–294, Feb. 2022.
[23] K. Li, Y. Wang, and M. Liu, "A non-cooperative game model for reliability-based task scheduling in cloud computing," *Int. J. Commun. Syst.*, vol. 33, no. 15, Oct. 2020, Art. no. e4512.
[24] A. Mämmelä, J. Riekki, A. Kotelba, and A. Anttonen, "Multidisciplinary and historical perspectives for developing intelligent and resource-efficient systems," *IEEE Access*, vol. 6, pp. 17464–17499, 2018.
[25] X. Zhao and C. Huang, "Microservice based computational offloading framework and cost efficient task scheduling algorithm in heterogeneous fog cloud network," *IEEE Access*, vol. 8, pp. 56680–56694, 2020.
[26] A. Lakhan, M. S. Memon, Q.-U.-A. Mastoi, M. Elhoseny, M. A. Mohammed, M. Qabulio, and M. Abdel-Basset, "Cost-efficient mobility offloading and task scheduling for microservices IoVT applications in container-based fog cloud network," *Cluster Comput.*, vol. 25, no. 3, pp. 2061–2083, Jun. 2022.
[27] V. Peláez, A. Campos, D. F. García, and J. Entrialgo, "Online scheduling of deadline-constrained bag-of-task workloads on hybrid clouds," *Concurrency Comput., Pract. Exper.*, vol. 30, no. 19, Oct. 2018, Art. no. e4639.
[28] Q.-U.-A. Mastoi, T. Ying Wah, R. Gopal Raj, and A. Lakhan, "A novel cost-efficient framework for critical heartbeat task scheduling using the Internet of Medical Things in a fog cloud system," *Sensors*, vol. 20, no. 2, p. 441, Jan. 2020.
[29] A. Singh, N. Auluck, O. Rana, A. Jones, and S. Nepal, "Scheduling real-time security aware tasks in fog networks," *IEEE Trans. Services Comput.*, vol. 14, no. 6, pp. 1981–1994, Nov. 2021.
[30] A. Hussain, M. Aleem, A. Khan, M. A. Iqbal, and M. A. Islam, "RALBA: A computation-aware load balancing scheduler for cloud computing," *Cluster Comput.*, vol. 21, no. 3, pp. 1667–1680, Sep. 2018.
[31] H. Yan, X. Zhu, H. Chen, H. Guo, W. Zhou, and W. Bao, "DEFT: Dynamic fault-tolerant elastic scheduling for tasks with uncertain runtime in cloud," *Inf. Sci.*, vol. 477, pp. 30–46, Mar. 2019.
[32] S. Nabi, M. Ibrahim, and J. M. Jimenez, "DRALBA: Dynamic and resource aware load balanced scheduling approach for cloud computing," *IEEE Access*, vol. 9, pp. 61283–61297, 2021.
[33] Y. Caniou, E. Caron, A. Kong Win Chang, and Y. Robert, "Budget-aware scheduling algorithms for scientific workflows with stochastic task weights on infrastructure as a service cloud platforms," *Concurrency Comput., Pract. Exper.*, vol. 33, no. 17, Sep. 2021, Art. no. e6065.
[34] S. Subbaraj and R. Thiyagarajan, "Performance oriented task-resource mapping and scheduling in fog computing environment," *Cognit. Syst. Res.*, vol. 70, pp. 40–50, Dec. 2021.
[35] E. Hosseini, M. Nickray, and S. Ghanbari, "Optimized task scheduling for cost-latency trade-off in mobile fog computing using fuzzy analytical hierarchy process," *Comput. Netw.*, vol. 206, Apr. 2022, Art. no. 108752.
[36] K. Li, "Scheduling parallel tasks with energy and time constraints on multiple manycore processors in a cloud computing environment," *Future Gener. Comput. Syst.*, vol. 82, pp. 591–605, May 2018.
[37] L. Liu, D. Qi, N. Zhou, and Y. Wu, "A task scheduling algorithm based on classification mining in fog computing environment," *Wireless Commun. Mobile Comput.*, vol. 2018, pp. 1–11, Aug. 2018.
[38] 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. 116218–116226, 2019.
[39] G. Sharma, N. Miglani, and A. Kumar, "PLB: A resilient and adaptive task scheduling scheme based on multi-queues for cloud environment," *Cluster Comput.*, vol. 24, no. 3, pp. 2615–2637, Sep. 2021.
[40] S. Azizi, M. Shojafar, J. Abawajy, and R. Buyya, "Deadline-aware and energy-efficient IoT task scheduling in fog computing systems: A semi-greedy approach," *J. Netw. Comput. Appl.*, vol. 201, May 2022, Art. no. 103333.
[41] S. Tuli, G. Casale, and N. R. Jennings, "GOSH: Task scheduling using deep surrogate models in fog computing environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 11, pp. 2821–2833, Nov. 2022.
[42] J. Luo, L. Yin, J. Hu, C. Wang, X. Liu, X. Fan, and H. Luo, "Container-based fog computing architecture and energy-balancing scheduling algorithm for energy IoT," *Future Gener. Comput. Syst.*, vol. 97, pp. 50–60, Aug. 2019.
[43] M. Lavanya, B. Shanthi, and S. Saravanan, "Multi objective task scheduling algorithm based on SLA and processing time suitable for cloud environment," *Comput. Commun.*, vol. 151, pp. 183–195, Feb. 2020.
[44] D. Tychalas and H. Karatza, "A scheduling algorithm for a fog computing system with bag-of-tasks jobs: Simulation and performance evaluation," *Simul. Model. Pract. Theory*, vol. 98, Jan. 2020, Art. no. 101982.
[45] P. Zhang, X. Ma, Y. Xiao, W. Li, and C. Lin, "Two-level task scheduling with multi-objectives in geo-distributed and large-scale SaaS cloud," *World Wide Web*, vol. 22, no. 6, pp. 2291–2319, Nov. 2019.
[46] H. Li, K. Ota, and M. Dong, "Deep reinforcement scheduling for mobile crowdsensing in fog computing," *ACM Trans. Internet Technol.*, vol. 19, no. 2, pp. 1–18, Apr. 2019.
[47] X. Li, Y. Qin, H. Zhou, D. Chen, S. Yang, and Z. Zhang, "An intelligent adaptive algorithm for servers balancing and tasks scheduling over mobile fog computing networks," *Wireless Commun. Mobile Comput.*, vol. 2020, pp. 1–16, Jul. 2020.
[48] D. Ding, X. Fan, Y. Zhao, K. Kang, Q. Yin, and J. Zeng, "Q-learning based dynamic task scheduling for energy-efficient cloud computing," *Future Gener. Comput. Syst.*, vol. 108, pp. 361–371, Jul. 2020.
[49] Z. Tong, F. Ye, B. Liu, J. Cai, and J. Mei, "DDQN-TS: A novel bi-objective intelligent scheduling algorithm in the cloud environment," *Neurocomputing*, vol. 455, pp. 419–430, Sep. 2021.
[50] B. Kruekaew and W. Kimpan, "Multi-objective task scheduling optimization for load balancing in cloud computing environment using hybrid artificial bee colony algorithm with reinforcement learning," *IEEE Access*, vol. 10, pp. 17803–17818, 2022.

[51] M. M. Razaq, S. Rahim, B. Tak, and L. Peng, "Fragmented task scheduling for load-balanced fog computing based on Q-learning," *Wireless Commun. Mobile Comput.*, vol. 2022, pp. 1–9, Mar. 2022.

[52] V. Sindhu, M. Prakash, and P. Mohan Kumar, "Energy-efficient task scheduling and resource allocation for improving the performance of a cloud–fog environment," *Symmetry*, vol. 14, no. 11, p. 2340, Nov. 2022.

[53] H. Wadhwa and R. Aron, "Optimized task scheduling and preemption for distributed resource management in fog-assisted IoT environment," *J. Supercomput.*, vol. 79, no. 2, pp. 2212–2250, Feb. 2023.

[54] M. B. Gawali and S. K. Shinde, "Task scheduling and resource allocation in cloud computing using a heuristic approach," *J. Cloud Comput.*, vol. 7, no. 1, pp. 1–16, Dec. 2018.

[55] L. Chen, X. Li, and R. Ruiz, "Idle block based methods for cloud workflow scheduling with preemptive and non-preemptive tasks," *Future Gener. Comput. Syst.*, vol. 89, pp. 659–669, Dec. 2018.

[56] L. Zuo, S. Dong, L. Shu, C. Zhu, and G. Han, "A multiqueue interlacing peak scheduling method based on Tasks' classification in cloud computing," *IEEE Syst. J.*, vol. 12, no. 2, pp. 1518–1530, Jun. 2018.

[57] S. K. Panda and P. K. Jana, "An energy-efficient task scheduling algorithm for heterogeneous cloud computing systems," *Cluster Comput.*, vol. 22, no. 2, pp. 509–527, Jun. 2019.

[58] W. Cai, J. Zhu, W. Bai, W. Lin, N. Zhou, and K. Li, "A cost saving and load balancing task scheduling model for computational biology in heterogeneous cloud datacenters," *J. Supercomput.*, vol. 76, no. 8, pp. 6113–6139, Aug. 2020.

[59] J. Liu, J. Ren, W. Dai, D. Zhang, P. Zhou, Y. Zhang, G. Min, and N. Najjari, "Online multi-workflow scheduling under uncertain task execution time in IaaS clouds," *IEEE Trans. Cloud Comput.*, vol. 9, no. 3, pp. 1180–1194, Jul. 2021.

[60] G. Yao, Q. Ren, X. Li, S. Zhao, and R. Ruiz, "A hybrid fault-tolerant scheduling for deadline-constrained tasks in cloud systems," *IEEE Trans. Services Comput.*, vol. 15, no. 3, pp. 1371–1384, May 2022.

[61] K. R. Remesh Babu and P. Samuel, "Service-level agreement–aware scheduling and load balancing of tasks in cloud," *Softw., Pract. Exper.*, vol. 49, no. 6, pp. 995–1012, Jun. 2019.

[62] M. Mukherjee, M. Guo, J. Lloret, R. Iqbal, and Q. Zhang, "Deadline-aware fair scheduling for offloaded tasks in fog computing with inter-fog dependency," *IEEE Commun. Lett.*, vol. 24, no. 2, pp. 307–311, Feb. 2020.

[63] R. M. Abdelmoneem, A. Benslimane, and E. Shaaban, "Mobility-aware task scheduling in cloud-fog IoT-based healthcare architectures," *Comput. Netw.*, vol. 179, Oct. 2020, Art. no. 107348.

[64] S. Arisdakessian, O. A. Wahab, A. Mourad, H. Otrok, and N. Kara, "FoGMatch: An intelligent multi-criteria IoT-fog scheduling approach using game theory," *IEEE/ACM Trans. Netw.*, vol. 28, no. 4, pp. 1779–1789, Aug. 2020.

[65] M. R. Hossain, M. Whaiduzzaman, A. Barros, S. R. Tuly, M. J. N. Mahi, S. Roy, C. Fidge, and R. Buyya, "A scheduling-based dynamic fog computing framework for augmenting resource utilization," *Simul. Model. Pract. Theory*, vol. 111, Sep. 2021, Art. no. 102336.

[66] M. Jia, J. Zhu, and H. Huang, "Energy and delay-ware massive task scheduling in fog-cloud computing system," *Peer-to-Peer Netw. Appl.*, vol. 14, no. 4, pp. 2139–2155, Jul. 2021.

[67] D. Rahbari, "Analyzing meta-heuristic algorithms for task scheduling in a fog-based IoT application," *Algorithms*, vol. 15, no. 11, p. 397, Oct. 2022.

[68] A. Khan, A. Abbas, H. A. Khattak, F. Rehman, I. U. Din, and S. Ali, "Effective task scheduling in critical fog applications," *Scientific Program.*, vol. 2022, pp. 1–15, Mar. 2022.

[69] J. Lim, "Latency-aware task scheduling for IoT applications based on artificial intelligence with partitioning in small-scale fog computing environments," *Sensors*, vol. 22, no. 19, p. 7326, Sep. 2022.

[70] J. Wan, B. Chen, S. Wang, M. Xia, D. Li, and C. Liu, "Fog computing for energy-aware load balancing and scheduling in smart factory," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4548–4556, Oct. 2018.

[71] J. P. B. Mapetu, Z. Chen, and L. Kong, "Low-time complexity and low-cost binary particle swarm optimization algorithm for task scheduling and load balancing in cloud computing," *Int. J. Speech Technol.*, vol. 49, no. 9, pp. 3308–3330, Sep. 2019.

[72] N. Mansouri, B. Mohammad Hasani Zade, and M. M. Javidi, "Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory," *Comput. Ind. Eng.*, vol. 130, pp. 597–633, Apr. 2019.

[73] M. Z. Hasan and H. Al-Rizzo, "Task scheduling in Internet of Things cloud environment using a robust particle swarm optimization," *Concurrency Comput., Pract. Exper.*, vol. 32, no. 2, Jan. 2020, Art. no. e5442.

[74] W. Jing, C. Zhao, Q. Miao, H. Song, and G. Chen, "QoS-DPSO: QoS-aware task scheduling for cloud computing system," *J. Netw. Syst. Manage.*, vol. 29, no. 1, pp. 1–29, Jan. 2021.

[75] S. Javanmardi, M. Shojafar, R. Mohammadi, A. Nazari, V. Persico, and A. Pescapè, "FUPE: A security driven task scheduling approach for SDN-based IoT–Fog networks," *J. Inf. Secur. Appl.*, vol. 60, Aug. 2021, Art. no. 102853.

[76] X. Tang, C. Shi, T. Deng, Z. Wu, and L. Yang, "Parallel random matrix particle swarm optimization scheduling algorithms with budget constraints on cloud computing systems," *Appl. Soft Comput.*, vol. 113, Dec. 2021, Art. no. 107914.

[77] S. Nabi, M. Ahmad, M. Ibrahim, and H. Hamam, "AdPSO: Adaptive PSO-based task scheduling approach for cloud computing," *Sensors*, vol. 22, no. 3, p. 920, Jan. 2022.

[78] A. M. Yadav, K. N. Tripathi, and S. C. Sharma, "An opposition-based hybrid evolutionary approach for task scheduling in fog computing network," *Arabian J. for Sci. Eng.*, vol. 48, no. 2, pp. 1547–1562, Feb. 2023.

[79] Y. Wang, C. Guo, and J. Yu, "Immune scheduling network based method for task scheduling in decentralized fog computing," *Wireless Commun. Mobile Comput.*, vol. 2018, pp. 1–8, Sep. 2018.

[80] X. Ma, H. Gao, H. Xu, and M. Bian, "An IoT-based task scheduling optimization scheme considering the deadline and cost-aware scientific workflow for cloud computing," *EURASIP J. Wireless Commun. Netw.*, vol. 2019, no. 1, Dec. 2019, Art. no. 249.

[81] B. M. Nguyen, H. Thi Thanh Binh, T. The Anh, and D. Bao Son, "Evolutionary algorithms to optimize task scheduling problem for the IoT based bag-of-tasks application in cloud–fog computing environment," *Appl. Sci.*, vol. 9, no. 9, p. 1730, Apr. 2019.

[82] P. Pirozmand, A. A. R. Hosseinabadi, M. Farrokhzad, M. Sadeghilalimi, S. Mirkamali, and A. Slowik, "Multi-objective hybrid genetic algorithm for task scheduling problem in cloud computing," *Neural Comput. Appl.*, vol. 33, no. 19, pp. 13075–13088, Oct. 2021.

[83] S. Velliangiri, P. Karthikeyan, V. M. Arul Xavier, and D. Baswaraj, "Hybrid electro search with genetic algorithm for task scheduling in cloud computing," *Ain Shams Eng. J.*, vol. 12, no. 1, pp. 631–639, Mar. 2021.

[84] M. Abdel-Basset, R. Mohamed, R. K. Chakrabortty, and M. J. Ryan, "IEGA: An improved elitism-based genetic algorithm for task scheduling problem in fog computing," *Int. J. Intell. Syst.*, vol. 36, no. 9, pp. 4592–4631, Sep. 2021.

[85] R. Li, "Use linear weighted genetic algorithm to optimize the scheduling of fog computing resources," *Complexity*, vol. 2021, Jun. 2021, Art. no. 9527430.

[86] F. Emara, "Genetic-based multi-objective task scheduling algorithm in cloud computing environment," *Int. J. Intell. Eng. Syst.*, vol. 14, no. 5, pp. 571–582, Oct. 2021.

[87] Z. Yu, "Research on optimization strategy of task scheduling software based on genetic algorithm in cloud computing environment," *Wireless Commun. Mobile Comput.*, vol. 2022, pp. 1–9, Jun. 2022.

[88] R. Zhou, "A heuristic task scheduling strategy for intelligent manufacturing in the big data-driven fog computing environment," *Mobile Inf. Syst.*, vol. 2022, pp. 1–10, Aug. 2022.

[89] M.-T. Zhou, T.-F. Ren, Z.-M. Dai, and X.-Y. Feng, "Task scheduling and resource balancing of fog computing in smart factory," *Mobile Netw. Appl.*, vol. 28, no. 1, pp. 19–30, Feb. 2023.

[90] R. Sing, S. K. Bhoi, N. Panigrahi, K. S. Sahoo, M. Bilal, and S. C. Shah, "EMCS: An energy-efficient makespan cost-aware scheduling algorithm using evolutionary learning approach for cloud-fog-based IoT applications," *Sustainability*, vol. 14, no. 22, p. 15096, Nov. 2022.

[91] A. Tarafdar, M. Debnath, S. Khatua, and R. K. Das, "Energy and makespan aware scheduling of deadline sensitive tasks in the cloud environment," *J. Grid Comput.*, vol. 19, no. 2, pp. 1–25, Jun. 2021.

[92] H. Baniata, A. Anaqreh, and A. Kertesz, "PF-BTS: A privacy-aware fog-enhanced blockchain-assisted task scheduling," *Inf. Process. Manage.*, vol. 58, no. 1, Jan. 2021, Art. no. 102393.

[93] E. Elsedimy and F. Algarni, "MOTS-ACO: An improved ant colony optimiser for multi-objective task scheduling optimisation problem in cloud data centres," *IET Netw.*, vol. 11, no. 2, pp. 43–57, Mar. 2022.

[94] M. S. Sanaj, P. M. J. Prathap, and V. Alappatt, "Profit maximization based task scheduling in hybrid clouds using whale optimization technique," *Inf. Secur. Journal: A Global Perspective*, vol. 29, no. 4, pp. 155–168, Jul. 2020, doi: 10.1080/19393555.2020.1716116.

[95] Z. Movahedi, B. Defude, and A. M. Hosseininia, "An efficient population-based multi-objective task scheduling approach in fog computing systems," *J. Cloud Comput.*, vol. 10, no. 1, pp. 1–16, Dec. 2021.

[96] S. Mangalampalli, S. K. Swain, and V. K. Mangalampalli, "Prioritized energy efficient task scheduling algorithm in cloud computing using whale optimization algorithm," *Wireless Pers. Commun.*, vol. 126, no. 3, pp. 2231–2247, Oct. 2022.

[97] A. Najafizadeh, A. Salajegheh, A. M. Rahmani, and A. Sahafi, "Privacy-preserving for the Internet of Things in multi-objective task scheduling in cloud-fog computing using goal programming approach," *Peer-to-Peer Netw. Appl.*, vol. 14, no. 6, pp. 3865–3890, Nov. 2021.

[98] A. Najafizadeh, A. Salajegheh, A. M. Rahmani, and A. Sahafi, "Multi-objective task scheduling in cloud-fog computing using goal programming approach," *Cluster Comput.*, vol. 25, no. 1, pp. 141–165, Feb. 2022.

[99] S. Ben Alla, H. Ben Alla, A. Touhafi, and A. Ezzati, "An efficient energy-aware tasks scheduling with deadline-constrained in cloud computing," *Computers*, vol. 8, no. 2, p. 46, Jun. 2019.

[100] 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," *Trans. Emerg. Telecommun. Technol.*, vol. 31, no. 2, Feb. 2020, Art. no. e3770.

[101] M. S. Sanaj and P. M. Joe Prathap, "Nature inspired chaotic squirrel search algorithm (CSSA) for multi objective task scheduling in an IAAS cloud computing atmosphere," *Eng. Sci. Technol., Int. J.*, vol. 23, no. 4, pp. 891–902, Aug. 2020.

[102] J. Prassanna and N. Venkataraman, "Adaptive regressive holt–winters workload prediction and firefly optimized lottery scheduling for load balancing in cloud," *Wireless Netw.*, vol. 27, no. 8, pp. 5597–5615, Nov. 2021.

[103] M. R. Shirani and F. Safi-Esfahani, "Dynamic scheduling of tasks in cloud computing applying dragonfly algorithm, biogeography-based optimization algorithm and Mexican hat wavelet," *J. Supercomput.*, vol. 77, no. 2, pp. 1214–1272, Feb. 2021.

[104] M. A. Elaziz and I. Attiya, "An improved Henry gas solubility optimization algorithm for task scheduling in cloud computing," *Artif. Intell. Rev.*, vol. 54, no. 5, pp. 3599–3637, Jun. 2021.

[105] A. M. Yadav, K. N. Tripathi, and S. C. Sharma, "An enhanced multi-objective fireworks algorithm for task scheduling in fog computing environment," *Cluster Comput.*, vol. 25, no. 2, pp. 983–998, Apr. 2022.

[106] K. Mishra, J. Pati, and S. Kumar Majhi, "A dynamic load scheduling in IaaS cloud using binary Jaya algorithm," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, no. 8, pp. 4914–4930, Sep. 2022.

[107] S. Ghanavati, J. Abawajy, and D. Izadi, "An energy aware task scheduling model using ant-mating optimization in fog computing environment," *IEEE Trans. Services Comput.*, vol. 15, no. 4, pp. 2007–2017, Jul. 2022.

[108] H. Ben Alla, S. Ben Alla, A. Touhafi, and A. Ezzati, "A novel task scheduling approach based on dynamic queues and hybrid meta-heuristic algorithms for cloud computing environment," *Cluster Comput.*, vol. 21, no. 4, pp. 1797–1820, Dec. 2018.

[109] 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, Feb. 2019.

[110] O. H. Ahmed, J. Lu, A. M. Ahmed, A. M. Rahmani, M. Hosseinzadeh, and M. Masdari, "Scheduling of scientific workflows in multi-fog environments using Markov models and a hybrid salp swarm algorithm," *IEEE Access*, vol. 8, pp. 189404–189422, 2020.

[111] A. Mubeen, M. Ibrahim, N. Bibi, M. Baz, H. Hamam, and O. Cheikhrouhou, "Alts: An adaptive load balanced task scheduling approach for cloud computing," *Processes*, vol. 9, no. 9, p. 1514, Aug. 2021.

[112] M. S. Ajmal, Z. Iqbal, F. Z. Khan, M. Ahmad, I. Ahmad, and B. B. Gupta, "Hybrid ant genetic algorithm for efficient task scheduling in cloud data centers," *Comput. Electr. Eng.*, vol. 95, Oct. 2021, Art. no. 107419.

[113] S. M. Hussain and G. R. Begh, "Hybrid heuristic algorithm for cost-efficient QoS aware task scheduling in fog–cloud environment," *J. Comput. Sci.*, vol. 64, Oct. 2022, Art. no. 101828.

[114] Z. Yin, F. Xu, Y. Li, C. Fan, F. Zhang, G. Han, and Y. Bi, "A multi-objective task scheduling strategy for intelligent production line based on cloud-fog computing," *Sensors*, vol. 22, no. 4, p. 1555, Feb. 2022.

[115] M. A. Elaziz, S. Xiong, K. P. N. Jayasena, and L. Li, "Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution," *Knowledge-Based Syst.*, vol. 169, pp. 39–52, Apr. 2019.

[116] P. Hosseinioun, M. Kheirabadi, S. R. Kamel Tabbakh, and R. Ghaemi, "A new energy-aware tasks scheduling approach in fog computing using hybrid meta-heuristic algorithm," *J. Parallel Distrib. Comput.*, vol. 143, pp. 88–96, Sep. 2020.

[117] M. Abd Elaziz, L. Abualigah, and I. Attiya, "Advanced optimization technique for scheduling IoT tasks in cloud-fog computing environments," *Future Gener. Comput. Syst.*, vol. 124, pp. 142–154, Nov. 2021.

[118] M. Nanjappan, G. Natesan, and P. Krishnadoss, "HFTO: Hybrid firebug tunicate optimizer for fault tolerance and dynamic task scheduling in cloud computing," *Wireless Pers. Commun.*, vol. 129, no. 1, pp. 323–344, Mar. 2023.

[119] I. Attiya, L. Abualigah, D. Elsadek, S. A. Chelloug, and M. Abd Elaziz, "An intelligent chimp optimizer for scheduling of IoT application tasks in fog computing," *Mathematics*, vol. 10, no. 7, p. 1100, Mar. 2022.

[120] P. Memari, S. S. Mohammadi, F. Jolai, and R. Tavakkoli-Moghaddam, "A latency-aware task scheduling algorithm for allocating virtual machines in a cost-effective and time-sensitive fog-cloud architecture," *J. Supercomput.*, vol. 78, no. 1, pp. 93–122, Jan. 2022.

[121] B. Mohammad Hasani Zade, N. Mansouri, and M. M. Javidi, "SAEA: A security-aware and energy-aware task scheduling strategy by parallel squirrel search algorithm in cloud environment," *Expert Syst. Appl.*, vol. 176, Aug. 2021, Art. no. 114915.

[122] S. I. AlShathri, S. A. Chelloug, and D. S. M. Hassan, "Parallel meta-heuristics for solving dynamic offloading in fog computing," *Mathematics*, vol. 10, no. 8, p. 1258, Apr. 2022.

[123] X. Chen and S. Xiao, "Multi-objective and parallel particle swarm optimization algorithm for container-based microservice scheduling," *Sensors*, vol. 21, no. 18, p. 6212, Sep. 2021.

[124] J. Luo, D. El Baz, R. Xue, and J. Hu, "Solving the dynamic energy aware job shop scheduling problem with the heterogeneous parallel genetic algorithm," *Future Gener. Comput. Syst.*, vol. 108, pp. 119–134, Jul. 2020.

[125] F. Luo, Y. Yuan, W. Ding, and H. Lu, "An improved particle swarm optimization algorithm based on adaptive weight for task scheduling in cloud computing," in *Proc. 2nd Int. Conf. Comput. Sci. Appl. Eng.*, Oct. 2018, pp. 1–5.

[126] R. Zhang, F. Tian, X. Ren, Y. Chen, K. Chao, R. Zhao, B. Dong, and W. Wang, "Associate multi-task scheduling algorithm based on self-adaptive inertia weight particle swarm optimization with disruption operator and chaos operator in cloud environment," *Service Oriented Comput. Appl.*, vol. 12, no. 2, pp. 87–94, Jun. 2018.

[127] X. Huang, C. Li, H. Chen, and D. An, "Task scheduling in cloud computing using particle swarm optimization with time varying inertia weight strategies," *Cluster Comput.*, vol. 23, no. 2, pp. 1137–1147, Jun. 2020.

[128] J. Zhang, J. Sheng, J. Lu, and L. Shen, "UCPSO: A uniform initialized particle swarm optimization algorithm with cosine inertia weight," *Comput. Intell. Neurosci.*, vol. 2021, pp. 1–18, Mar. 2021.

[129] C. E. Andrade, T. Silva, and L. S. Pessoa, "Minimizing flowtime in a flowshop scheduling problem with a biased random-key genetic algorithm," *Expert Syst. Appl.*, vol. 128, pp. 67–80, Aug. 2019.

[130] K. S. Subramanian and G. Teshite, "Performance evaluation of novel random biased-genetic algorithm (NRB-GA): A hybrid load balancing algorithm in a cloud computing environment," *Sci. Program.*, vol. 2022, pp. 1–13, Dec. 2022.

[131] N. Manikandan, N. Gobalakrishnan, and K. Pradeep, "Bee optimization based random double adaptive whale optimization model for task scheduling in cloud computing environment," *Comput. Commun.*, vol. 187, pp. 35–44, Apr. 2022.

ZULFIQAR ALI KHAN received the Bachelor of Science degree (Hons.) in information technology from the Virtual University of Pakistan and the Master of Science degree in computing from the Shaheed Zulfikar Ali Bhutto Institute of Science and Technology (SZABIST), Islamabad. He is currently pursuing the Ph.D. degree with Universiti Teknologi PETRONAS (UTP), Malaysia. His research interests include task scheduling, cloud computing, optimization, fog computing, meta-heuristic algorithms, and the Internet of Things (IoT).

IZZATDIN ABDUL AZIZ received the Ph.D. degree in information technology from Deakin University, Australia, working in the domain of hydrocarbon exploration and cloud computing. He is currently an Associate Professor with Universiti Teknologi PETRONAS (UTP), Malaysia. He is also heading the Center for Research in Data Science (CeRDaS), solving complex upstream and downstream oil and gas (O&G) industry problems utilizing machine learning and big data analytics. He is also working closely with O&G companies in delivering solutions for complex problems, such as offshore O&G pipeline corrosion rate prediction, O&G pipeline corrosion detection, rotating machines, process failure prediction, securing data on clouds, and bridging upstream and downstream oil and gas businesses through data analytics. Additionally, he is also working on fundamental computer science problems, such as algorithm's optimization.

NURUL AIDA BT OSMAN was with the Artificial Intelligence Research Laboratory, MIMOS Berhad. She is currently a Lecturer with Universiti Teknologi PETRONAS and also a Researcher with the Centre of Research in Data Science (CeRDaS). Her research interests include machine learning, predictive analytics, recommender systems, ontology development, and sentiment analysis.

ISRAR ULLAH received the M.S. degree in computer science from the National University of Computer and Emerging Sciences (NUCES), Islamabad, Pakistan, in 2009, and the Ph.D. degree in computer engineering from Jeju National University, South Korea, in 2019. His research interests include the application of prediction and optimization algorithms to the build IoT-based solutions, analytical modeling, network simulation, and analysis of optimization algorithms.