International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(1s) 2026 pp. 1-37
https://doi.org/10.70917/ijcisim-2026-2023
Article

A Systematic Review on Task Scheduling in Cloud Computing

Kavita Rani[1], Om Prakash Sangwan[2], Ritu Garg[3]

[1] Department of Computer Science and Engineering, Guru Jambheshwar University of Science and Technology (GJUS&T), Hisar, Haryana, India. k.ghanghas67@gmail.com
[2] Department of Computer Science and Engineering, Guru Jambheshwar University of Science and Technology (GJUS&T), Hisar, Haryana, India. sangwan0863@gmail.com
[3] Department of Computer Engineering, National Institute of Technology (NIT) Kurukshetra, Haryana, India. ritu.59@gmail.com

Abstract: Cloud computing provides on-demand access to virtualized resources, enabling efficient task execution and service delivery. Among its core components, task scheduling plays a crucial role in optimizing resource utilization, reducing execution time, and enhancing overall system performance. This systematic literature review examines 58 research articles published between 2018 and June 2025, focusing on scheduling techniques in cloud computing. The review categorizes existing approaches—including traditional, machine learning-based, heuristic, metaheuristic, and hybrid methods—while analyzing their objectives, considered factors, and reported limitations. Findings reveal that although significant advancements have been achieved, key challenges remain in ensuring availability, reliability, adaptability, and energy efficiency, particularly under dynamic and heterogeneous cloud environments. Many proposed methods demonstrate strong results in controlled experiments but struggle with scalability in real-world deployments. Moreover, issues such as uncertainty handling, SLA compliance, fault tolerance, and sustainability are insufficiently addressed. The review highlights the adaptability of existing algorithms and their potential for integration with emerging technologies like artificial intelligence, edge computing, and the Internet of Things (IoT). Future research opportunities lie in developing intelligent, multi-objective, and context-aware scheduling strategies that balance cost, performance, energy consumption, and security. Overall, task scheduling in cloud computing remains a complex yet promising area of study, with significant potential for innovation to meet evolving user and system demands.
Keywords: Reliability, Fault Tolerance, Non-Positional Number System, Residual Classes, Computer System.

1. Introduction

In terms of technological advancement, cloud computing represents a revolutionary shift, providing users with unmatched access to resources from any location at any time, all while following a usage-driven payment model. Fundamentally, this invention differs from the conventional method of depending on users' local computer hard drives by storing and processing data and applications online. A shared infrastructure, flexible pay-as-you-go pricing, strong scalability, and remote accessibility are all necessary for this idea to become a reality. Virtualizing computational units, exposing them to dynamic configurations, and utilizing the advantages of economic scale are all areas in which cloud computing shines. Individuals, small and medium-sized businesses (SMEs), and government organizations are increasingly adopting this technology as the go-to platform for handling both ordinary and mission-critical applications [1]. The unique features of cloud computing, including administration, agility, continuous availability, delegated maintenance, disaster recovery capabilities, elasticity, economic efficiency, on-demand services, operational expenditure considerations, performance optimization, reliability assurance, scalability, strong security protocols, and fostering user trust, are responsible for this adoption boom. The model allows for the sharing of resources, such as servers, storage, and apps, which may be quickly supplied and released with little administration work or in response to service provider interactions.

The history of cloud computing and its numerous advancements can be traced back to the time when interconnected computers introduced distributed computing, which in turn gave rise to cluster computing, grid

computing, and cloud computing. By offering scalable and elastic access to resources through a pay-as-use model, cloud computing removes the need for companies and organizations to invest in purchasing hardware and software resources. Massive computing power is needed to complete several applications from large-scale scientific fields, such as data mining, business informatics, physics, astronomy, and bioinformatics, in a reasonable amount of time. In order to meet the increasing computing demands of customers, large-scale applications have recently been moved to the cloud. The flexibility and elasticity of cloud resources, which are easily accessible to pay-as-you-go cloud users, are responsible for this. Large data volumes are processed by cloud computing, and if the jobs are not scheduled correctly, performance may suffer. Effective task scheduling has a direct effect on how well the cloud computing paradigm works [2].

Scheduling algorithms are crucial for job organization, as they provide substantial advantages and enhance performance. Although resource utilization tends to improve CPU performance, concurrent application execution can reduce CPU throughput. Distributed computing is used in many operations, including data transfer and administrative work, and it can be effectively managed to cut expenses. In order to maximize efficiency and cost, these jobs are scheduled according to client requests. Task scheduling gets more complicated as more clients use cloud services, which calls for efficient scheduling algorithms. The purpose of task scheduling in cloud computing systems is to identify alternate methods of allocating conflicting cloud jobs to scarce resources in a way that maximizes one or more goals. Current cloud scheduling techniques are widely believed to have their roots in the early days of the grid scheduling paradigm or distributed system principles. Although cloud computing makes it possible for users to manually employ large amounts of virtualized resources, work distribution is still difficult. At the virtual machine layer, virtualization and commercialization help manage these challenges. Efficient resource allocation to tasks depends on efficient scheduling. Numerous methods are frequently employed, such as task heuristic-based, cloud service-based, workflow-based, and task dynamics-based scheduling. Effective and efficient scheduling is constantly required in order to properly manage and oversee the cloud's diverse infrastructures [3].

2. Overview Of Cloud Computing

Cloud computing is built around clear service and deployment models that shape how it operates. The three main service models are Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). These models are used to provide resources on the Internet. SaaS enables users to connect to and use cloud-based apps over the internet whereas users do not need to install or maintain the software application and hardware. SaaS can be found in such industries and applications as Google Apps, Salesforce, Office 365, and Netflix. PaaS allows users to create and launch applications using tools, programming languages, and services provided by vendors, while the infrastructure remains outside user control [4]. Noteworthy PaaS providers are Google App Engine and Microsoft Azure. IaaS acts as the backbone of cloud computing by offering virtualized computing resources, which users can manage regarding operating systems, storage, and applications but not the main infrastructure. Popular IaaS providers include AWS, Rackspace Open Cloud, and IBM Smart Cloud Enterprise. Cloud computing is based upon four deployment models apart from service models (i.e., private, public, community and hybrid clouds). Private clouds focus on security and control for a particular organization. Public clouds are accessible to anyone and operated by a third party. Community clouds serve organizations with common goals, compliance needs, or technical requirements, while hybrid clouds combine different infrastructures to allow flexibility and resource sharing among private, public, and community systems. In this setting, effective task scheduling poses a major challenge due to the changing nature of resources and varied user needs. Cloud service providers must manage a wide range of requests, with some requiring little cost and resources while others need significant computational power and bandwidth. To meet these needs, task scheduling mechanisms are crucial for optimizing resource use, ensuring fairness, and improving user satisfaction. By prioritizing tasks based on factors such as bandwidth availability, execution time, cost, and reliability, cloud providers can enhance both operational efficiency and quality of service (QoS) [5].

3. Task Scheduling

Scheduling is an important process. It involves deciding which tasks to perform during specific time periods. The goal is to optimize the flow of data across one or more objects. In cloud computing, a cloud broker manages scheduling to allocate specific times for various tasks. In a distributed system, scheduling means assigning tasks to resources and coordinating their execution while keeping dependencies intact. This mapping aims to meet different user-defined Quality of Service (QoS) requirements. These QoS parameters usually relate to performance metrics like execution time and non-functional needs such as security and energy use. In cloud computing, scheduling is essential for efficient resource use [6]. Improper scheduling can lead to underutilization, bottlenecks, and decreased

performance since many users share cloud resources with different needs. Effective scheduling makes it easier to allocate resources like CPU, memory, storage, and bandwidth according to user priorities and workload characteristics. One primary objective of cloud-based task scheduling is to reduce makespan, which is the time required to complete all tasks. Lowering makespan ensures timely service delivery and increases throughput. Another important goal is loading balancing, which distributes work evenly among available resources. This prevents some nodes from being overwhelmed while others sit idle. It reduces the chance of performance problems and helps maintain system stability. Scheduling decisions also play a key role in energy efficiency. Data centers consume a lot of power [7].

Energy-aware scheduling algorithms have been created to minimize idle resource time, consolidate workloads, and cut costs. Cost-efficient scheduling ensures that tasks are assigned to resources in a way that keeps expenses low for both cloud service providers and their customers while meeting necessary QoS requirements. Additionally, scheduling in cloud environments must consider different hardware and the need to scale. Unlike traditional computing systems, cloud infrastructures include data centers spread across various locations, virtual machines with different processing powers, and workloads that change dynamically. A good scheduling mechanism adapts to this variety while ensuring reliability and consistency in service. Security and fault tolerance are also important factors. Scheduling must protect data confidentiality, integrity, and availability, especially for sensitive applications [8]. Fault-tolerant scheduling ensures that tasks can be rescheduled on different resources if failures occur, without disrupting the overall workflow. Many scheduling strategies have been proposed in the literature, ranging from static to dynamic approaches. Static scheduling assigns tasks before execution starts. This works well for predictable workloads. Dynamic scheduling adjusts to changes in task arrival patterns and resource availability in real time. Metaheuristic algorithms like Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO) are also commonly used for complex scheduling situations. They can find near-optimal solutions in a reasonable amount of time. Overall, task scheduling in cloud computing is not just a technical need. It plays a crucial role in service efficiency, cost-effectiveness, and user satisfaction [9]. By using performance-focused, energy-aware, and QoS-driven scheduling methods, cloud systems can provide reliable services in various application areas such as healthcare, finance, education, and scientific research.

4. Need Of Task Scheduling

The task scheduling is one of the most important parts in the cloud computing resource management. As cloud environments host different applications and heterogeneous users with different requirements, efficient task scheduling leads to the effective resource utilization, high performance, and QoS improvement. Below are some good reasons why task scheduling is important in cloud environments:

- Efficient Resource Utilization: Task scheduling ensures that computing resources (such as CPU, memory, storage, and network bandwidth) are utilized efficiently for multiple tasks. Without scheduling, some resources would may remain idle while others become overloaded resulting in diminished system throughput. Proper scheduling algorithm helps in reducing this fall by distributing the load equally and utilizing resources in an efficient manner [10].
- Improved Quality of Service (QoS): QoS is a primary concern in cloud environments, which includes response time, throughput, and makespan. User request is assured to be completed in the required time limits from the task scheduling. Scheduling directly impacts deadlines and the SLA (service-level agreement) that providers promised to users by reducing response time.
- Load Balancing and Scalability: As the cloud environment is filled with more and more tasks and users, load balancing thus becomes a necessity in order to keep the performance at the desired level. Scheduling algorithms assign the workload evenly to different virtual machines (VMs), thus ensuring that there are no servers that are overloaded with a high number of requests and, as a result, bottlenecks. In addition, this increases scalability because the system can now handle varying capacities without losing its effectiveness [11].
- Cost and Energy Efficiency: Service providers in the cloud are aiming at cutting down on expenses needed during operations. Efficient scheduling makes the process greener by reducing the amount of energy consumed for the resources that are in the idle state. Additionally, through the implementation of scheduling, the principle of using the most economically configured resource for carrying out the tasks is observed; thus, both infrastructure and operational costs are reduced.

- Reliability and Dynamic Resource Allocation: The environment of the cloud is so volatile, user demands can go up and down, and resources can be available at one time and not at the next. Task scheduling allows the shifting of resources instantly to wherever it is most needed to assure reliability even if the loads are unpredictable. The feature of this adaptability assures that any failure or change in resource provision will not have a negative impact on the overall system performance [12].

5. The Model And Objective Function

A submission (application), a target computation architecture, and scheduling standards make up the scheduling structure model. A problem can be represented as a $(DAG) = G\ (T, E, R, C)$ as shown in figure 1

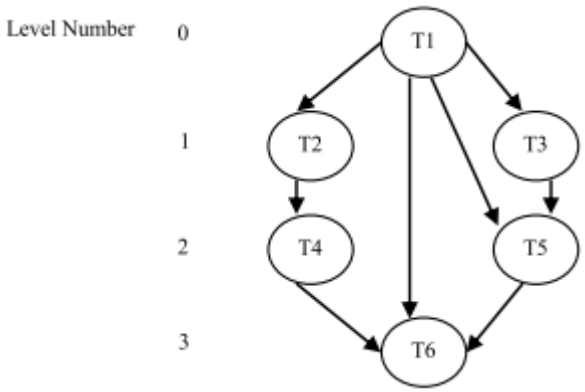

Fig.1. A model of DAG

where n tasks are represented by $T = \{t_i, i = 0, 1, 2, \ldots, n-1\}$ [28–30]. $e_{i,j}$ denotes the precedence limits between two connected tasks, and symbol E denotes a set of edges between tasks $E = \{e_{i,j}, i < j\}$. These interconnected tasks, tasks $t_i, t_j \in T$ indicate that task t_j operation is dependent on task t_i, hence limiting its precedence. It shows that the outcomes of task t_i will be used as the input value for task t_j, and that t_j cannot start implementing it before t_i. The task t_i is the predecessor of t_j, while t_j is the heir of t_i . The $2D$ matrix of size $v \times m$ is represented by R in this case, and the estimated operation time of v_i on the j^{th} processor is indicated by r_{ij} in R. If two tasks are t_i and t_j, the communication cost [13] between them is represented by the matrix $CmC(t \times t)$. In the graph below, an entry task is one that has no ancestors, and an exit (leave) task is one that has no descendants. A cloud framework is made up of a set $VM = \{vm_i where i = 0, 1, 2, \ldots m - 1\}$ of m self-regulatory virtual machines (VMs) connected by a high-rate network, as shown in Fig. 2.

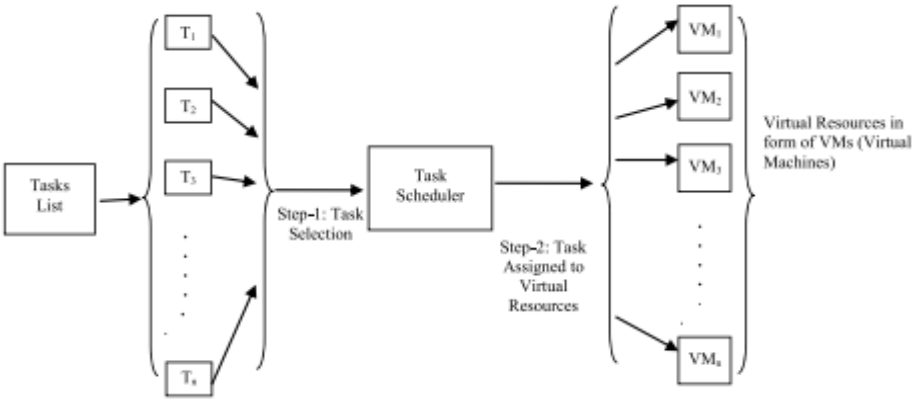

Fig.2. Task scheduling in a cloud-based framework

The cloud architecture's altered network bandwidth may create changes in the Data Transfer Frequency (DTF). A two-dimensional array of size $m \times m\ or\ DTFm \times m$ between any two virtual machines can be used to represent DTF. To perform a task $t_i$on a virtual machine $VM\ vm_j$, the Probable Execution Cost (PEC) can be represented by an additional $2D$ array $PECn \times m$, where $0 <= i <= n - 1$ and $0 <= j <= m - 1$. The

PEC might vary for each virtual machine and builds upon its computational speed. Two factors determine the communication cost between vm_x and vm_y. The installed frequency of the processors on both communication sides comes first. The second is the correspondence cost value of the frequency [14]. They assume that there is no transmission channel conflict and that every VM workstation can send data to other VM workstations. They also assume there is no communication cost between jobs scheduled on identical virtual machines. The goal of the task arrangement challenge is to arrange all of the tasks of a particular submission to machines in a way that minimizes the application's completion time while meeting all precedence constraints.

6. Phases Of Task Scheduling

In cloud computing, scheduling is essential for efficient resource use [6]. Improper This is an optimization problem of matching user tasks to cloud resources with various objectives such as efficiency, cost and reliability. In cloud computing, task scheduling is important to carry out workloads that meet performance objectives, budget constraints, and SLAs. Different phases of task scheduling are discussed in figure 3.

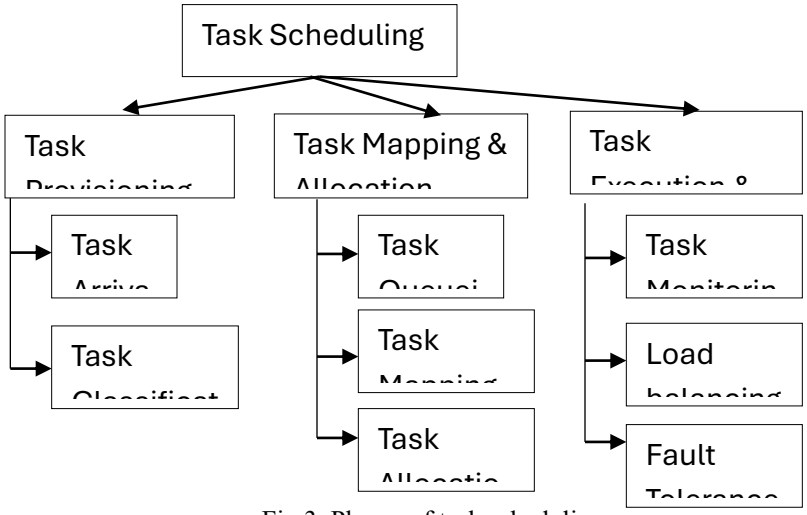

Fig.3. Phases of task scheduling

I. Task Provisioning

In the definition of the term fault tolerance there are three main aspects of its use.

- Task Arrival: Tasks arrival period always exists where new task shall be presented by the users and compete to enter the scheduling system to be taken care off. Different workloads, including batch jobs and real-time applications, must be processed by the scheduler.
- Task Classification: The new tasks are categorized based on multiple parameters like priority, sensitivity to deadlines, and resource requirements (e.g., CPU, memory, or I/O bound).
- Task Queues: Once classified, tasks are enqueued in different dedicated queues such as priority queues for high-priority tasks, deadline queues for tasks having a due date, and batch queues for background tasks. Queue management is important for the scheduler to keep in order, and avoid starvation of jobs.

II. Task Mapping & Allocation

This phase involves the mapping and allocation of jobs to the best resources on the cloud

- Task Mapping: It is the logical mapping of pending tasks to available resources (load balancer, storage of cloud) including physical servers, virtual machines and containers. This phase takes in consideration elements such as resource capacity, compatibility, and distance.
- Task Allocations: After scheduling, the scheduled task is assigned to available resources, optimizing cost, performance, and QoS. Allocation approaches may be static and dynamic as well, and the heuristics and metaheuristics that are applied could be heuristic and metaheuristic, for workload and building type considerations.

III. Task Execution & Management

During this stage, the planned tasks are performed persistently and efficiently.

- Task Monitoring: Continuous monitoring is performed to track execution progress, resource utilization, and performance metrics. This allows early detection of bottlenecks or anomalies.
- Load Balancing: To prevent resource overload, the system redistributes tasks dynamically across underutilized resources. Effective load balancing improves overall system throughput and reduces execution time.
- Fault Tolerance: Rescheduling, job duplication, and checkpointing are examples of fault-tolerant techniques that are used in the event of node failures or resource unavailability. This minimizes interruptions, assures task completion even in the event of failure, and ensures dependability.

7. Types Of Task Scheduling

Task scheduling in the cloud requires more care than it does in traditional methods because of its crucial significance in cloud services. Efficient task scheduling is crucial since cloud operations and resource use are quite expensive. Given the variety and unique characteristics of cloud resources, work allocation requires careful thought. A key factor in improving the adaptability and dependability of cloud-based frameworks is task scheduling. Finding the appropriate order for tasks to be completed is the main goal in order to provide the finest results for clients within the allotted time constraints [15]. The sequence and requirements of jobs and their subtasks in cloud computing affect how resources are gradually distributed among different components, including networks, firewalls, and containers. This makes cloud task scheduling a significant difficulty because there is no guarantee that a preset order will hold true while the project is being prepared. Given the simultaneous availability of numerous tasks and the unpredictability of resource availability, task flow, and execution approaches, task scheduling must be dynamic. Typically, there are two types of task scheduling:

- Static Task Scheduling: Static task scheduling jobs are assigned to nodes before job execution, and the processing node is chosen during compilation. The work continues until it is finished when resources are assigned. Reduced runtime overhead scheduling and minimal node and processor utilization are the main goals of static task planning. However, there are drawbacks to this strategy. It ignores an application's particular workloads and resource requirements, which may lead to ineffective resource use and possible job failures.
- Dynamic Task Scheduling: In contrast to the static approach, dynamic task scheduling involves making decisions as the work is being executed. Among other things, considerations include energy efficiency, intra- and inter-process traffic, and resources. In accordance with cluster resource availability and application demand, it also makes task migration easier. Efficient use of resources, low energy consumption, and timely task completion are the main goals of dynamic task planning [16]

8. Research Objectives

The following are the stated research objectives

- To examine and compile the latest and most sophisticated cutting-edge work scheduling strategies in cloud computing over the previous four years.
- To give researchers advice on how to choose relevant datasets, test settings, and metrics for their work in the field by summarizing the metrics, datasets, and test environments utilized in the evaluated publications

By offering a thorough summary of the most recent developments in the quickly developing field, this work seeks to expand our understanding of task scheduling in cloud computing and meet the aforementioned goals.

8.1. Sections of the Article

The organization of this work is presented across sixteen sections for clarity and coherence. The study begins with the Background to establish the context, followed by an Overview of Cloud Computing and an Overview of Task Scheduling to highlight the fundamental concepts. The Need of Task Scheduling in Cloud Environment is then discussed to underline its significance. Next, the Model and Objective Function is introduced, along with the Phases of Task Scheduling and the Types of Task Scheduling, which provide insight into the process and approaches used.

The Research Objectives are then stated to define the scope and direction of the work. A detailed account of Related Work is presented, supported by the Objectives of Task Scheduling and an Overview of Scheduling Constraints in Cloud Computing to explore the challenges and goals associated with scheduling. The methodology is explained through sections on Testing Tools and Datasets and Research Techniques, which describe the resources and methods adopted in this study. The Results and Discussion section follows, providing analysis and interpretation of findings. Finally, the work concludes with a section on Limitations & Future Research Issues, leading to the Conclusion, which summarizes the contributions and key outcomes of the study.

9. Related Work

Scheduling of tasks in cloud computing is a major research area, and over time researchers have been suggesting several different kinds of approaches to improve system performance, utilization of resources, and energy saving. Most of the current works can be categorized according to the following classes: traditional methods (emphasizing rule-based scheduling policies), machine learning-based approaches (making use of predictive models), heuristic techniques (developed specifically for the optimization of the problem at hand), metaheuristic algorithms (which provide a global search of the solution space), and hybrid methods (which integrate several strategies to overcome the shortcomings of individual ones). Such a classification allows for a structured analysis of approaches by their strengths and weaknesses.

9.1 Traditional Approaches for Cloud Task Scheduling

K. J. Reddy et al. (2025) analyzed how to map user requests with resource availability using the Resource-Aware Scheduling (RAS) method, which speeds up resource allocation [21]. The suggested technique dynamically distributed resources by looking at log data on workload behavior, resource availability, cost, job efficiency, and decreased waste. It did this by mapping user data with resource data and scheduling according to priority and availability. In a private cloud environment, the simulation-based experiment illustrated resource allocation, utilization, response time, and cost. To assess the RAS's performance, it was also contrasted with the conventional scheduling algorithm. According to the evaluation, the RAS model worked better for allocating cloud resources. Y. Pachipala et al. (2024) presented a novel Modified Shortest Job First (SJF) algorithm in the CloudSim simulation framework, which led to a revolutionary approach to job scheduling in cloud systems [22]. Addressing current issues with conventional scheduling algorithms, reducing resource bottlenecks, shortening job completion times, and enhancing system efficiency were the main goals of this research. When used in a CloudSim context, the SJF algorithm showed encouraging results in terms of improved resource usage, higher throughput, and reduced makespan. Researchers investigated how task prioritizing mechanisms affected SJF performance and explored dynamic SJF variations that adapted to shifting workload situations.

M. Prashanthi et al. (2024) investigated the challenges of scheduling tasks in dynamic cloud systems, focusing on workload management and energy efficiency optimization [23]. The difficulties presented by dynamic, large-scale cloud systems were frequently too much for traditional heuristic-based methods to handle. The potential of energy-efficient scheduling techniques to lower energy usage without sacrificing system performance was examined in the paper. Additionally, it investigated sophisticated scheduling strategies such as optimization algorithms, machine learning-based approaches, and hybrid heuristics. The study sought to provide creative ways to optimize resource use, reduce energy expenses, and raise the general effectiveness of cloud task scheduling by critically evaluating the status of the field and identifying important obstacles. I. S. Aref, et al. (2022) suggested a hybrid task scheduling method that minimized Makespan and maximized resource load balances by combining three algorithms: Min-Min scheduling, Max-Min scheduling, and genetic algorithm [24]. Genetic algorithms were essential to this work because they helped determine which tasks should be planned using the Min-Min strategy and which should be scheduled using the Max-Min algorithm. The effectiveness of the suggested task is evaluated using two metrics: Makespan and resource consumption. When compared to the Min-Min and Max-Min scheduling algorithms, the experimental findings demonstrated that the suggested approach produced the best results in terms of efficiently minimizing the Makespan and utilizing all available resources.

Y. Bo et al. (2022) suggested a dynamic priority and scheduling decision (TS-DP)-based task scheduling method for cloud computing [25]. Based on the task value and task urgency, the algorithm determined the task dynamic priority. Simultaneously, in order to mimic the behavior of fireflies, the decision variables for task scheduling were determined by attraction (the user's anticipated earliest completion time), load balance, and fluorescence brightness (task processing time constraint). The tasks were then scheduled to the executable computing node that corresponded to the maximum decision variable value based on task priority. According to

experimental results, the technique may efficiently enhance cloud computing's resource scheduling performance, reduce task completion times, and provide good load balancing—particularly when there are a lot of tasks and nodes.

S. Choudhary et al. (2022) focused on the resource management-based tasks where the executive devices may uphold the task on a work space [26]. Task scheduling was a key concept in all functional systems that enabled the system to support multitasking and other features. In this study, the suggested Enhanced Round Robin (ERR) scheduling method was presented and assessed. The suggested approach assisted in decreasing the task migration and task completion time by accounting for delayed product details such as bandwidth, storage, processing capacity, and task deadline. The performance of the suggested algorithm was examined by comparing it to the existing methods. M. S. Sanaj et al. (2020) suggested an enhanced version of the round robin algorithm (ERR) that increased efficiency and performed better without compromising the positive aspects of the original RR [27]. Using the CloudSim toolkit, the suggested algorithm was put into practice and tested. The preliminary findings demonstrated that, under the identical circumstances, the average waiting time for the jobs in a specific number of cloudlets was lower in MRR than in conventional RR. In terms of execution time and residue energy, the suggested approach likewise performed better than the other algorithms already in use, including ACO, GA, MPA, Min-Min, and PSO. M. Karaja et al. (2020) created a multi-cloud system in which clouds were connected to meet the needs of clients [28]. The variability of resources in these situations made task scheduling extremely difficult. A dynamic Bag-of-Tasks scheduling approach for a heterogeneous multi-cloud environment with a limited budget was proposed in this study. They proved the algorithm's usefulness in terms of makespan by conducting experiments on proposed synthetic data sets.

A. K. M. Rahman Mazumder, et al. (2019) created numerous scheduling methods. Nevertheless, the majority of created algorithms had their own constraints and competing goals [29]. As a result, one was effective at one kind of task but ineffective at another. In this study, they suggested a method that minimized competing objectives and distributed the algorithms based on the job type. First Come First Serve (FCFS), Shortest Job First (SJF), Round Robin (RR), Max-Min, and Min-Min were among the algorithms they examined. Lastly, they demonstrated their suggested approach, which balanced the trade-off between makespan and average waiting time while reducing the drawbacks of other algorithms. Y. Yu et al. (2019) suggested a TQ (Three Queues) algorithm for cloud task scheduling that uses three queues and dynamic priority [30]. The TQ algorithm initially ranked jobs in the waiting queue based on their priority before classifying them into different job types based on the amount of data input and output from the Map phase, the total number of active node tasks, the completion time of Map tasks, and the disk I/O rate. Jobs were grouped into queues to maximize the use of hardware. The findings of the experiment demonstrated that the technique could efficiently enhance cloud task scheduling performance when CPU-intensive and I/O-intensive workloads coexisted, and it could also reduce the task's overall completion time. Y. T. H. Hlaing et al. (2019) suggested a Static Independent Task Scheduling on Virtualized Servers in a Cloud Computing Environment, where tasks were assigned to the appropriate virtual machine (VM) based on the number of available processing elements, cost, and processing power of each resource, and by classifying tasks based on the length of their instructions [31]. When compared to scheduling algorithms like Shortest Job First (SJF) and First Come First Serve (FCFS), the results of this method's simulation using Cloud Simulator (CloudSim toolkit) demonstrated that it maximized overall execution time and minimized execution cost for all tasks. S. Kang et al. (2018) suggested a unique multi-cloud system design that met the intricate needs of users' applications [32]. In order to get excellent performance, they suggested a dynamic scheduling strategy (DSS) based on this architecture that combined node availability prediction methods with the Divisible Load Theory. To test the effectiveness of the suggested scheduling approach, they run extensive simulations. The findings demonstrated that the suggested scheduling approach reduced the overall processing time of loads by up to 44.60%, outperforming the baseline schemes. The outcomes also offered helpful information on how well the suggested strategy worked in practical situations.

Table 1. Comparative Analysis of Traditional Task scheduling approaches

Author & Year	Technique Used	Dataset	Performance Metric	Simulation Tool	Findings	Limitations
K. J. Reddy et al. (2025)	Resource-Aware Scheduling (RAS)	Not specified	Resource allocation, utilization, response time, cost	CloudSim	Dynamically distributed resources using	Focused only on private cloud; scalability in heterogeneous

					workload logs; improved resource allocation compared to conventional scheduling	clouds not addressed
Y. Pachipala et al. (2024)	Modified Shortest Job First (SJF)	Not specified	Resource usage, throughput, makespan	CloudSim	Improved resource usage, higher throughput, reduced makespan	Performance may degrade under unpredictable workloads
M. Prashanthi et al. (2024)	Energy-efficient scheduling using optimization, ML, and hybrid heuristics	Google Cloud dataset	Energy consumption, resource utilization, performance	Cloudsim	Identified potential of energy-efficient and intelligent scheduling methods	Lacked real-world validation; mostly conceptual
I. S. Aref et al. (2022)	Hybrid (Min-Min + Max-Min + Genetic Algorithm)	Not specified	Makespan, resource consumption	Not specified	Minimized makespan, improved load balance, better than Min-Min and Max-Min	Higher computational complexity due to hybrid approach
Y. Bo et al. (2022)	TS-DP (Task Scheduling with Dynamic Priority and Firefly Algorithm)	Benchmark dataset	Task completion time, load balance, scheduling efficiency	WorkflowSim	Improved scheduling performance, reduced task completion times, balanced load in large-scale tasks	Complexity increases with number of tasks and nodes
S. Choudhary et al. (2022)	Enhanced Round Robin (ERR)	Not specified	Task migration, completion time	CloudSim	Reduced migration and completion time, considered bandwidth, storage, and deadlines	Limited evaluation against modern metaheuristics
M. S. Sanaj et al. (2020)	Enhanced Round Robin (ERR)	Google Cloud dataset	Waiting time, execution time, energy consumption	CloudSim	Reduced waiting time, improved execution and energy efficiency	Limited to synthetic scenarios; scalability not tested

					over ACO, GA, PSO, Min-Min	
M. Karaja et al. (2020)	Bag-of-Tasks Scheduling in Multi-Cloud	Not specified	Makespan	Not specified	Dynamic scheduling in budget-limited environments improved makespan	Did not evaluate energy or cost efficiency
A. K. M. Rahman Mazumder et al. (2019)	Comparative analysis of FCFS, SJF, RR, Min-Min, Max-Min	Not specified	Makespan, average waiting time	CloudSim	Proposed trade-off-based method balancing makespan and waiting time	No focus on energy consumption or dynamic workloads
Y. Yu et al. (2019)	TQ (Three Queues) Algorithm	Google Cloud dataset	Completion time, scheduling efficiency	CloudSim	Improved scheduling for mixed CPU- and I/O-intensive workloads; reduced completion time	Complexity due to multi-queue priority management
Y. T. H. Hlaing et al. (2019)	Static Independent Task Scheduling on Virtualized Servers	Google Cloud dataset	Execution time, execution cost	CloudSim	Improved execution time and minimized cost over SJF, FCFS	Static nature unsuitable for dynamic workloads
S. Kang et al. (2018)	Dynamic Scheduling Strategy (DSS) using Divisible Load Theory	Google Cloud dataset	Processing time of loads	CloudSim	Reduced processing time by 44.6%, outperformed baseline methods	Needs validation in real-world large-scale clouds

9.2 Machine Learning Approaches for Cloud Task Scheduling

J. Wang et al. (2025) suggested a Soft Actor-Critic (RTPA-SAC) algorithm-based real-time performance-aware job scheduling technique [33]. This technique improved load balancing by improving environmental consistency and flexibility in stochastic, dynamic job scheduling by dynamically detecting changes in server load performance in real-time. To assess task execution efficiency, they first built a constrained load performance loss function while taking parallel task interference into account. An incentive system that considered both load variations and response times was then implemented. It optimized task load variance while adhering to quality-of-service requirements in order to reduce response time. Lastly, the suggested scheduling approach improved exploratory and stable decision-making in task scheduling by utilizing the Soft Actor-Critic algorithm. As demonstrated by improvements in task response time, average task load variance, and task success rate, the experimental results demonstrated that RTPA-SAC performed better in load balancing than baseline techniquesJ. Anand, et al. (2025) proposed Efficiency-Aware Adjustable Deep Reinforcement Learning (EADRL), a system that made intelligent context-aware scheduling decisions possible by introducing two essential mechanisms: a dynamic confidence-aware reward adjustment and an

adjustable learning rate [34]. Over time, the learning rate adjusted to reward patterns, enhancing convergence while preserving stability under various load scenarios. Experiments showed that EADRL outperformed current benchmark techniques, such as heuristic-based techniques like Best-Fit, Random, and Earliest Idle Time First (EITF), as well as DRL-based techniques like Double DQN (DDQN), Deep Deterministic Policy Gradient (DDPG), and Server Real-Time Performance Deep Reinforcement Learning (SRP-DRL). Important parameters including task response time, success rate, and load variance showed these gains. The outcomes confirmed that EADRL is appropriate for scalable and latency-sensitive edge-cloud systems since it can adaptively optimize scheduling policies without knowing ahead of time what workloads will be required. H. Hou et al. (2024) suggested an improved task scheduling method based on the DQN framework that addressed the drawbacks of DQN and reduced the overestimation bias by using a Double DQN strategy and a more optimized Dueling-network design [35]. In order to overcome the issue of low utilization brought on by uniform sampling from replay memory, it additionally included a prioritized experience replay technique to accomplish importance sampling of experience data. According to experimental results, EETS outperformed both DQN and DDQN in terms of rewards and convergence rates. Key parameters such as energy usage, average task response time, and average machine working time were among the scheduling performance indicators where EETS performed better than other baseline algorithms. It had a major advantage, especially when managing big batches of work.

Xinglong Pei, et al. (2024) suggested DeepMIC, a multi-resource interleaving technique for task scheduling in cloud-edge systems that is based on deep reinforcement learning (DRL) [36]. In order to minimize the weighted-sum delay of the outstanding jobs, they first developed a multi-resource queueing approach. Based on the network data gathered via Software-Defined Networking (SDN) and the Mobile Edge Computing (MEC) management framework, the suggested model took into account demands for computing, caching, and forwarding resources within a node. Then, in order to accommodate the increased work flow, they tailored a DRL algorithm to guarantee a prompt model solution. Lastly, they showed that DeepMIC improved resource efficiency and decreased the average task response time by arranging the jobs in a flexible manner. Jiahui Pan et al. (2024) provided a dynamic online scheduling framework for several real-time workflows [37]. A deep reinforcement learning-based algorithm (R-DQN) was used by the scheduler in this framework to allocate each task in workflows to an appropriate virtual machine instance depending on the circumstances at hand. Its objective was to maximize resource usage and minimize workflow makespan by achieving efficient job scheduling and resource distribution. In addition to conducting experiments utilizing a variety of workflow benchmarks with varying structures and scales on the WorkflowSim platform, they offered a thorough description of the design and implementation of their approach. In terms of makespan and other metrics, the trial findings showed that their method performed noticeably better than the compared algorithms. Additionally, their algorithm performed well in terms of scalability, robustness, and generality.

D. Changzhi et al. (2024) created a simulation program for cloud work scheduling that uses multi-agent deep reinforcement learning [38]. Through real-time interactive learning and environmental feedback, the software achieved dynamic and adaptive allocation of cloud computing resources, fully utilizing the autonomy and cooperative features of multi-agent systems. The Python platform, on which the software was built, allowed for more effective adaptive methods, parameter adjustments, and a decrease in the average maximum job completion time for resource allocation and task scheduling issues in the dynamic cloud computing environment. Huanhuan Hou et al. (2024) conducted a quantitative comparison of the model differences of Markov Decision Process elements, examined energy consumption models used for scheduling purposes, and gave a summary of the DRL algorithms utilized in the literature [39]. The experimental platforms, datasets, and neural network architectures that were employed in the DRL algorithm were also compiled by them. The research gap in DRL-based task scheduling was finally examined, and current issues and potential future approaches were covered from a variety of angles. In addition to offering a reference for further investigation into the direction of DRL-based task scheduling research, this work advanced the correlation perspective on the task scheduling problem using the DRL approached. Their research revealed that DRL-based scheduling strategies have the potential to drastically lower cloud data center energy usage, which makes them an intriguing topic for future study.

A. Jayanetti et al. (2024) created a multi-agent RL framework to optimize workflow executions over multi-cloud settings in terms of green energy consumption [40]. Extensive simulation results showed that the suggested method performed 47% better than the comparison algorithms in terms of reducing the energy consumption of process executions while maintaining a makespan comparable to the comparison algorithms. Moreover, compared to a general multi-agent approach, the multi-agent technique learned five times faster with the suggested adjustments. N. K. Pandey et al. (2024) employed a deep reinforcement learning (RL)-based scheduling approach that took into

account all scheduling problems with various resources as targets and automatically learnt the best tactics that might produce optimal outcomes [41]. The outcomes demonstrated that the suggested approach, the RL method, was highly equivalent to the old strategy, flexible, communicative, and learnt tactics later on. X. Wang et al. (2023) suggested a brand-new offline DRL scheduling technique that preserved DRL's initial benefits while resolving the online trial-and-error problem [42]. They began by outlining the CMfg-SPs system model and putting forth the sequential Markov decision process modeling approach, in which every task was viewed as a single agent. The decision-making challenge in online scheduling was then transformed into an offline classification problem by introducing the decision transformer (DT) paradigm. Lastly, using the DT architecture, they trained an attention-based model offline as the agent's policy. According to experimental results, the suggested approach regularly matched or outperformed online DRL algorithms in terms of scheduling performance and model generalization, including proximal policy optimization (PPO), deep double Q-network (DDQN), deep recurrent Q-network (DRQN), and the offline learning algorithm behavior cloning (BC).

H. Mahmoud et al. (2022) suggested a decision tree-based multi-objective work scheduling algorithm for a heterogeneous setting [43]. They presented a novel technique for assigning and carrying out an application's task called Task Scheduling-Decision Tree (TS-DT). Heterogeneous Earliest Finish Time (HEFT), Technique for Order of Preference by Similarity to Ideal Solution, which included the Entropy Weight Method (TOPSIS-EWM), and combining Q-Learning with the Heterogeneous Earliest Finish Time (QL-HEFT) were the existing algorithms compared in order to assess the performance of the proposed TS-DT algorithm. According to their findings, the suggested TS-DT algorithm performed better than the current HEFT, TOPSIS-EWM, and QL-HEFT algorithms by, on average, lowering makespan by 5.21%, 2.54%, and 3.32%; increasing resource utilization by 4.69%, 6.81%, and 8.27%; and enhancing load balancing by 33.36%, 19.69%, and 59.06%. X. Wang et al. (2022) created a new scheduling method to address a dynamic scheduling issue in the cloud manufacturing setting for group services [44]. Multi-agent reinforcement learning was used to formulate and train their proposal. The recurrent neural network captured the processing trajectories of each task, while the graph convolution network stored the graph-structure properties of the tasks. Under the centralized training decentralized execution architecture, they trained the algorithm using a mixing network and separately created the action space and reward function. Graph convolution networks and multi-agent reinforcement learning were rarely applied to scheduling issues in cloud manufacturing. Their solution beat the other six multi-agent reinforcement learning-based scheduling algorithms in terms of generalizability and scheduling performance, according to contrast tests conducted on a case study. L. Ran et al. (2019) presented a deep reinforcement learning-based task scheduling technique and could dynamically allocate submitted jobs to various virtual machines. [45]. To identify the best job assignment solution that complies with Service Level Agreements (SLAs), they first defined the task scheduling as a dynamical optimum problem with constraints. They then used the deep deterministic policy gradients (DDPG) network. Without any prior knowledge, it relied solely on its experience to choose the best scheduling option. When faced with dynamic workloads, the suggested task scheduling method could successfully lower the average response time of tasks while ensuring load balancing among VMs, according to experimental results using real Alibaba cloud workload tracings.

Table 2. Comparative Analysis of Machine learning based Task scheduling approaches

Author & Year	Technique Used	Dataset	Performance Metric	Simulation Tool	Findings	Limitations
J. Wang et al. (2025)	Real-Time Performance-Aware Task Scheduling using Soft Actor-Critic (RTPA-SAC)	Google Cloud dataset	Task response time, task load variance, task success rate	Python Toolbox	Improved adaptability, stable decision-making, better load balancing	Computationally expensive; performance may degrade with large-scale tasks
J. Anand et al. (2025)	Efficiency-Aware Adaptive Deep RL (EADRL) with adaptive	Not specified	Response time, success rate, load variance	CloudSimPlus	Significant improvement over DRL & heuristic baselines; stable under varying	Requires careful reward design; sensitive to hyperparameter tuning

	learning rate & confidence-aware rewards				load	
H. Hou et al. (2024)	Energy-Efficient Task Scheduling (EETS) with Double DQN, Dueling network & prioritized replay	Google Cloud dataset	Energy consumption, response time, machine working time	Python Toolbox	Faster convergence; higher rewards than DQN & DDQN	May not scale well with large heterogeneous tasks
Xinglong Pei et al. (2024)	DeepMIC: DRL-based multi-resource interleaving with SDN & MEC	Not specified	Task response time, throughput, resource utilization	Matlab with CloudSim	Better resource utilization, lower average response time	Requires integration with SDN/MEC, increasing complexity
Jiahui Pan et al. (2024)	Online scheduling using R-DQN for workflow tasks	WorkflowSim benchmark dataset	Makespan, resource utilization	WorkflowSim	Outperforms compared algorithms in makespan & scalability	Workflow-specific; less generalizable to all task types
D. Changzhi et al. (2024)	Multi-Agent Deep RL-based cloud scheduling simulation software	Not specified	Completion time, adaptive strategy efficiency	Python Toolbox	Achieves adaptive allocation, lowers task completion time	Focused on simulation; lacks large-scale validation
Huanhuan Hou et al. (2024)	DRL-based scheduling analysis (survey) with MDP models	Google Cloud dataset	Energy consumption, QoS factors	CloudSim	Summarizes DRL methods; identifies energy saving benefits	Mainly a survey; lacks experimental validation
A. Jayanetti et al. (2024)	Multi-Agent RL for green energy-aware workflow scheduling	Not specified	Energy consumption, makespan	CloudSimPlus	Reduced energy by 47%, learned 5× faster	Needs more real-time deployment validation
N. K. Pandey et al. (2024)	Deep RL scheduling strategy for heterogeneou	Not specified	Task performance, adaptability	Matlab with CloudSim	Learns optimized strategies adaptively	Scalability challenges in large clouds

	s resources					
X. Wang et al. (2023)	Offline DRL scheduling with Decision Transformer & attention model	Google Cloud dataset	Scheduling performance, generalization	Python Toolbox	Matches/exceeds online DRL; avoids trial-and-error	Offline training costly; less reactive to dynamic changes
H. Mahmoud et al. (2022)	TS-DT (Decision Tree) for heterogeneous scheduling	Not specified	Makespan, resource utilization, load balancing	CloudSim	Outperforms HEFT, TOPSIS-EWM & QL-HEFT	May struggle with dynamic & stochastic environments
X. Wang et al. (2022)	Multi-agent RL with Graph Convolution & RNN	Google Cloud dataset	Scheduling performance, generalizability	Matlab with CloudSim	Outperforms six RL algorithms in case study	Computationally heavy; limited real-world tests
L. Ran et al. (2019)	DRL with DDPG for task scheduling	Alibaba Cloud workload traces	Response time, load balancing	CloudSim with CloudSim Plus	Reduced response time, improved VM load balancing	Relies on historical traces; adaptability to unseen workloads limited

9.3 Heuristic-Approaches for Cloud Task Scheduling

S. Chawla et al. (2024) suggested a fault-tolerant heuristic work scheduling approach to maximize cloud computing environments' use of resources [46]. In order to minimize makespan, balance load across resources, and provide fault tolerance, the algorithm used both replication and migration approaches. Under a variety of fault-prone conditions, simulation experiments showed that the algorithm improved resource efficiency by up to 13% when compared to benchmark methods. The outcomes demonstrated how well the suggested algorithm worked to improve resource efficiency and maintain performance in the event of resource breakdowns. W. Chongdarakul et al. (2024) presented Collision-Avoid based on Cloud Task Scheduling (CACS), a heuristic workflow scheduling algorithm that assigns client workflow represented by a directed acyclic graph (DAG) and executes the best cloud-fog nodes under the single client I/O port restriction to achieve the lowest possible delay, the shortest workflow completion time, and the best possible resource performance [47]. The simulation of the cloud-fog computing system was done using FogWorkflowSim. The testing findings demonstrated that the suggested approach outperformed the other compared algorithms in terms of scheduling results. Boroumandp et al. (2024) proposed a heuristic task scheduling technique for optimizing TSO-MCR (makespan-cost-reliability) goals. Furthermore, the suggested optimization model took into account the limitations of the consumers [48]. In order to do this, the task ranking approach, Pareto dominance, crowding distance, and disregarding the unreliable processors were used to acquire the trade-off between possibly competing objectives. The results of the simulation demonstrated that the suggested TSO-MCR performed noticeably better than other cutting-edge techniques. Its improvements over all equivalents in all 12 situations were 4.23, 8.93, 2.08, and 4.24%, respectively, in terms of reliability, makespan, total monetary cost, and the final score function that took into account all weighted objectives. Notably, the comparison was conducted using datasets with varying communication-to-computation ratio (CCR) values, covering both random applications and scientific procedures.

S. Lipsa et al. (2023) presented a parallel task scheduling method that, depending on whether a task is preemptive or non-preemptive, carried out heap construction and priority assignment to tasks simultaneously [49]. A suitable number of activities were included in the step-by-step illustrations of the suggested work. To ascertain the effectiveness of the suggested algorithms, the performance of the model was contrasted with some of the current methods, such as BATS, IDEA, and BATS+BAR, in terms of total waiting time and CPU time. To further illustrate

the task scheduling method's ability to manage jobs with varying priority, three additional scenarios were taken into consideration. Additionally, a dynamic cloud computing environment was used to test the work scheduling method. R. NoorianTalouki et al. (2022) suggested a novel approach to task prioritization and used techniques for task duplication to address the issue of dependent task scheduling in heterogeneous cloud computing systems [50]. Presenting a new list scheduling algorithm with a new task priority approach and using relevant job duplication techniques were the paper's novelties. In order to prioritize tasks in an efficient ordered list, this paper used the optimistic cost table downward (OCTd) and optimistic cost table upward (OCTu) procedures. Then, it performed task duplication using the Heterogeneous Earliest Finish Time (HEFT)-duplication method, which greatly decreased makespan. The suggested scheduling technique was empirically examined using several scientific workflows, including the Montage, FFT, Molecular, and LU-Like datasets, in order to validate the concept. The suggested technique outperformed other existing methods in terms of makespan, speedup, SLR, and efficiency—all of which were important scheduling assessment metrics—according to a performance comparison of the unique heuristic scheduling algorithm.

K. M. S. U. Bandaranayake et al. (2020) suggested TRETA (Total Resource Execution Time Aware Algorithm). The method found an ideal schedule by taking into account the overall execution time of the computing resource [51]. The algorithm was evaluated on real-world workload traces of NASA Ames iPSC/860 and compared with the state-of-the-art heuristics Min-Min, Min-Max, FCFS, and MCT for Makespan, Degree of Imbalance, and System Throughput. Using the CloudSim 5.0 Simulator, the experiment was conducted for workloads that comprised a larger collection of jobs. Comparing the suggested approach to other heuristics currently in use, it demonstrated a notable improvement in Makespan, Degree of Imbalance, and System Throughput. K. P. N. Jayasena, et al. (2020) suggested the Total Resource Execution Time Aware Algorithm (TRETA), a novel task scheduling algorithm that included the entire execution time of computing resources to provide an ideal schedule [52]. The algorithm's Makespan, Degree of Imbalance, and System Throughput were compared to the Min-Min, Min-Max, FCFS, and MCT heuristics. When compared to alternative algorithms, the suggested approach shown a notable improvement in Makespan. Better workload distribution among cloud resources was the outcome of the algorithm's superior performance over previous heuristics in terms of System Throughput and Degree of Imbalance. Weiwei Lin, et al. (2018) offered a cloud server power efficiency model [53]. In order to optimize energy usage in cloud environments, a heuristic task scheduling method (ECOTS) was created, and task scheduling was guided by server power efficiency. To lower system energy consumption at the lowest possible performance cost, ECOTS considered a number of important parameters, including job resource requirements, server power efficiency model, and performance degradation. The best scheduling plan could be approached more easily because to the ECOTS algorithm's superior global searching capabilities and reduced time and space complexity. To assess the efficacy of ECOTS, experiments were carried out in a heterogeneous cluster environment simulation. According to simulation results, the ECOTS algorithm was the most energy-efficient without going against the resource requirements of any cloud workload. Furthermore, when compared to the Improvement of Min-Min algorithm, the ECOTS algorithm successfully decreased overall energy consumption by over 20%.

C. Sudhakar et al. (2018) presented a multi-point crossover operator and an intelligent selection operator. In order to select superior heuristics, the intelligent selection operator penalized slower heuristics with a time weight [54]. Two solutions were combined using the multipoint crossover operator to provide new, varied, and potentially better solutions. The suggested method was put into practice in CloudSim and contrasted with the other accepted techniques. It was found that the suggested approach outperformed other conventional algorithms by 10.67% to 20.75% for a wide range of tasks. S. Devipriya et al. (2018) suggested a straightforward Max-min algorithm adjustment. The RASA algorithm and the Max-min strategy notion served as the foundation for this algorithm's development [55]. To surpass the RASA scheduling process in terms of total complete time for all submitted jobs, an improved Max-min algorithm was created. Rather than using entire time, the suggested Max-min approach was based on predicted execution time. Therefore, use Improved Max-min instead of original Max-min to schedule jobs in a cloud context resulted in a shorter makespan.

Table 3. Comparative Analysis of Heuristic based Task scheduling approaches

Author & Year	Technique Used	Dataset	Performance Metrics	Simulation Tool	Findings	Limitations
S. Chawla et al. (2024)	Fault-tolerant heuristic	Not specified	Resource utilization, makespan,	CloudSim	Improved utilization by 13%	Fault model limited, may

	scheduling with replication & migration		load balancing		under fault-prone scenarios; sustains performance during failures	not generalize to highly heterogeneous environments
W. Chongdarakul et al. (2024)	Collision-Avoid based Cloud Task Scheduling (CACS) for workflow DAGs in cloud-fog	Not specified	Workflow completion time, delay, resource performance	FogWorkflowSim	Achieved better scheduling results compared to other fog-cloud algorithms	Limited to single client I/O restriction scenarios
Boroumand dp et al. (2024)	Multi-objective heuristic TSO-MCR (makespan-cost-reliability) with Pareto dominance	Scientific workflow + random applications with different CCR	Makespan, cost, reliability, weighted score	CloudSim	Outperforms MOHEFT, CMSWC, HDCSA, MOBFD with 2–9% gains across 12 scenarios	Complexity increases with multi-objective trade-offs
S. Lipsa et al. (2023)	Parallel scheduling algorithm with priority assignment & heap building	Google Cloud dataset	Waiting time, CPU time, performance in dynamic environment	CloudSim	Improved waiting & CPU time over BATS, IDEA, BATS+BAR; effective under dynamic cloud workloads	Scalability to large cloud workloads not validated
R. NoorianTaloukai et al. (2022)	List scheduling with new task priority & duplication (OCTd/OCTu, HEFT-duplication)	Scientific workflows (Molecular, LU-Like, FFT, Montage)	Makespan, speedup, SLR, efficiency	WorkflowSim	Reduced makespan and improved efficiency vs. traditional HEFT	Dependent tasks only; may not suit independent task scheduling
K. M. S. U. Bandaranayake et al. (2020)	Total Resource Execution Time Aware Algorithm (TRETA)	NASA Ames iPSC/860 workload traces	Makespan, degree of imbalance, throughput	CloudSim 5.0	Significant improvement over Min-Min, Max-Min, FCFS, MCT	Focused only on execution time; energy not considered
K. P. N. Jayasena et al.	TRETA algorithm	Google Cloud	Makespan,	CloudSim	Improved	Similar limitation as

(2020)	(execution time aware)	dataset	throughput, degree of imbalance		makespan, throughput, and workload distribution	above; redundancy with TRETA study
W. Lin et al. (2018)	ECOTS heuristic (energy-aware scheduling using power efficiency model)	Google Cloud dataset	Energy consumption, task completion, performance degradation	Simulated heterogeneous cluster	Reduced energy consumption by >20% compared to Improved Min-Min	Tested in limited heterogeneous settings
C. Sudhakar et al. (2018)	Heuristic selection with intelligent operator & multi-point crossover	Google Cloud dataset	Makespan, task efficiency	CloudSim	Achieved 10–20% better performance for large number of tasks	May not scale well with very large workflows
S. Devipriya et al. (2018)	Improved Max-Min heuristic (based on RASA + Max-Min strategy)	Google Cloud dataset	Makespan, completion time	CloudSim	Reduced makespan compared to original Max-Min	Simplified model; limited evaluation with modern workflows

9.4 Meta-heuristic Approaches for Cloud Task scheduling

B. M. Hasani Zade et al. (2025) created the metaheuristic technique known as Beluga Whale Optimization (BWO). Nevertheless, even with an operator that increased population diversity, this approach still experienced local minima stagnation. In order to solve the primary drawbacks of traditional BWO, such as sluggish convergence and local optima traps, Opposition-Based Learning (OBL) was coupled with a Levy Fight Distribution (LFD) and a hybrid balance factor [56]. In order to address task scheduling issues that consider costs and makespan, a multi-objective version of improved BWO (IBWO) was suggested. The performance of MO-IBWO-Ring as a task scheduler was assessed in two scenarios: 1) using the Heterogeneous Computing Scheduling Problem (HCSP) as the benchmark dataset, which had 512 small tasks and 1024 medium tasks, and 2) using tasks and virtual machines that were produced at random. The suggested approach outperformed other approaches when evaluating provider metrics. Jeng-Shyang Pan et al. (2025) presented the Advanced Willow Catkin Optimization (AWCO) algorithm, an enhanced version. By using a quasi-opposition-based learning technique to improve the algorithm's global search capabilities and sinusoidal mapping to speed up its convergence, AWCO improved the algorithm's performance [57]. A thorough analysis using the CEC2014 benchmark suite, which included 30 test functions, showed that AWCO produced better optimization results than both traditional WCO and some well-known meta-heuristics. The suggested method also took into account trade-offs between the load balancing, makespan, and cost goals. When AWCO's experimental findings were contrasted with those derived from the other meta-heuristics, it became clear that the suggested algorithm performed better in task scheduling. The approach provided a strong basis for optimizing cloud computing resources in the area of task scheduling in a cloud computing setting.

Yan Han (2025) suggested a particle swarm optimization algorithm-based approach for cloud computing task scheduling, and experiments were conducted to examine the algorithm's impact on task completion time and energy usage in a cloud computing environment [58]. In order to satisfy customers' resource needs, the cloud computing platform's workloads were judiciously distributed using an intelligent scheduling technique. In addition to satisfying user demands, an effective cloud task scheduling approach should have maximized the usage of cloud computing resources and minimized energy consumption. However, the efficiency and quality of the results of the standard cloud task scheduling algorithm were subpar, and it was primarily accomplished through user involvement and

expertise. Cebrail Barut et al. (2024) suggested a rule-based, interpretable way to reduce this issue. This approach used a two-phase process, semi-offline and online, and combined machine learning approaches with metaheuristic work scheduling solutions [59]. Initially, metaheuristic solutions to task-scheduling problems that had been encountered or generated at random were archived during the semi-offline phase. The fundamental idea behind the suggested approach was to use these previously discovered effective metaheuristic solution patterns to comparable issues in the future. Using machine learning approaches, rule sets were automatically extracted from this large archive dataset in order to find common solution patterns. These interpretable rule sets were utilized to determine the best solution pattern and to determine the kind of task scheduling issue that arose during the online phase. This approach was tested and confirmed to work well in a variety of cloud task-scheduling scenarios, significantly cutting down on execution time and allowing the usage of metaheuristics in time-sensitive applications.

P. Tamilarasu et al. (2024) proposed an Improved Coati Optimization Algorithm-based Task Scheduling (ICOATS) to solve the problems of increased workload on Virtual Machines (VMs) in cloud computing environments, longer scheduling times, and excessive cost consumption [60]. In this suggested ICOATS, a model for work allocation and scheduling was constructed using the variables of virtual machines, cost, and time. Additionally, it featured a multi-objective fitness function that aimed to enhance resource utilization while simultaneously minimizing makespan. When compared to the current metaheuristic task scheduling methods, the simulation results of this ICOATS technique shown a greater improvement in makespan reduction while also improving turnaround efficiency, success rate, and availability. Laith Abualigah et al. (2024) suggested an improved optimization technique designed specifically for cloud-based work scheduling [61]. The strategy used a Levy flight mechanism to optimize exploration-exploitation trade-offs and accelerate convergence, building on the Jaya algorithm and Synergistic Swarm Optimization (SSO). Furthermore, by adding stochasticity to the search process through Levy flights, the algorithm was better able to avoid local optima and traverse intricate solution spaces. Results from experiments showed that the method was superior in terms of scalability, convergence speed, and solution quality. All things considered, the suggested Improved Jaya Synergistic Swarm Optimization Algorithm provided a viable way to optimize TSCC, which helped to improve system performance and resource usage in cloud-based applications. The suggested approach outperformed the original method by 10% and attained an overall accuracy of 88%.

P. Pabitha et al. (2024) suggested the Chameleon and Remora Search Optimization Algorithm (CRSOA), which explores the effects of MIPS and network bandwidth that directly affect the performance of virtual machines (VMs), in order to achieve an efficient scheduling procedure [62]. Furthermore, the work concurrently contained the uncertainty components of makespan, load balance, task completion rate, and scheduling cost during the scheduling process. comparison to other competitive metaheuristic task scheduling algorithms, the simulation results clearly demonstrated that the suggested CRSOA approach was lowering the completion time and handling the load balancing amongst the available VMs well. When compared to baseline techniques with varying numbers of tasks and virtual machines, the experimental study of this CRSOA shown its dominance in decreasing the makespan by 18.96%, cost by 22.18%, and degree of imbalance by 20.54%. Chirag Chandrashekar et al. (2024) suggested a Modified Chimp-Whale Optimization Algorithm (MCWOA) that combines aspects of the Whale Optimization Algorithm (WOA) and the Chimp Optimization Algorithm (COA) [63]. Furthermore, the whale optimization algorithm's bubble-net hunting and random search mechanisms were integrated into MCWOA's position-updating procedure, enhancing the original MCWOA's local search capabilities. According to simulation results, the new approach performed better than the original MCWOA, particularly in situations involving multiple damage detection. Energy consumption, computational cost, work time, and delay were among the measures used to evaluate the effectiveness of the suggested MCWOA. According to the simulated data, the new MCWOA performed better than the other approaches on every metric. The article also referenced the Whale Optimization Algorithm (WOA), Chimp Algorithm (CA), Grey Wolf Optimizer (GWO), Ant Lion Optimizer (ALO), and Genetic Algorithm (GA).

Z. A. Khan et al. (2024) suggested a parallel enhanced whale optimization technique for cloud scheduling of separate activities with different resources [64]. By employing an adaptive bubble net assaulting mechanism and a modified encircling maneuver, the suggested approach enhanced solution diversity and prevented local optima. Even though the parallelization technique was sophisticated inside, it maintained a short execution time. Throughput and resource usage were enhanced while the makespan was reduced by the suggested algorithm. In comparison to Multi-Core Random Matrix Particle Swarm Optimization (MRMPSO) and the top-performing enhanced whale optimization algorithm (WOAmM), it proved how successful the suggested PEWOA was. Better scalability was demonstrated by the algorithm's consistent improvement in performance across a range of task counts on the GoCJ dataset. A range of GoCJ and HCSP instances were used in the studies, which were carried out in CloudSim. A number of statistical tests were also performed to assess the results' significance. Z. Cui, et al. (2023) proposed a

multi-objective cloud work scheduling approach based on an evolutionary multi-factorial optimization algorithm [65]. To build a multi-objective cloud task scheduling model, execution time, execution cost, and virtual machine load balancing were selected as the objective functions. An assisted optimization job was created by combining the multi-objective multi-factor optimization (MO-MFO) algorithm with the task scheduling features, and then applying the multi-factor optimization (MFO) technique to the task scheduling problem. Simulation tests using the cloud task test dataset demonstrated that the approach greatly increased scheduling effectiveness. By employing a multi-factor approach and knowledge transfer to share the convergence direction among sub-populations, the scheduling method outperformed other evolutionary algorithms in finding the optimal solution interval and achieving the best results across all objective functions.

Sudheer Mangalampalli et al. (2023) created a multi-objective trust-aware scheduler that prioritized tasks, virtual machines, and scheduled tasks to allocate virtual resources in a way that minimizes energy usage and makespan [66]. To model the job scheduler, the Whale Optimization Algorithm was employed. The simulation was run entirely using CloudSim. Both simulated and real-time worklogs from NASA and HPC2N were part of the simulation's workload. The suggested method was contrasted with the current metaheuristic methods, such as ACO, GA, and PSO. According to simulation results, makespan, energy consumption, overall running time, and trust parameters—such as availability, success rate, and turnaround efficiency—all significantly improved. H. Zhang et al. (2023) examined the goals of cloud computing tasks and developed a multi-objective task scheduling model with system load and execution time as scheduling goals [67]. For the answer, the Cat Swarm Optimization model was presented, taking into account the difficulty of multi-objective task scheduling. Local optimization was a natural fit for the Cat Swarm Optimization concept. By modifying the weight component and fitness level, this model was enhanced. It outperformed the other two optimization models at this time, with an optimal optimization value of 0.506. The execution time of cloud computing was evaluated in the examination of cloud computing instances. The proposed model tended to converge after 102 iterations and a task execution time of 6.23 seconds. The research material offered crucial technical assistance for cloud computing resource management and scheduling optimization.

Raja Masadeh et al. (2021) suggested a cloud computing job scheduling method based on the multiple-objective model and Sea Lion Optimization Algorithm (SLnO) [68]. It made it possible to maximize resource use while reducing overall completion time, cost, and power consumption. The SLnO scheduler outperformed other state-of-the-art schedulers in terms of makespan, cost, energy consumption, resource usage, and degree of imbalance, according to the simulation findings on the tested data. VWOA, WOA, GWO, and RR were used to compare SLnO's performance across a range of activities and virtual machines. In comparison to the other algorithms, the results demonstrated that SLnO reduced makespan, imbalance degree, and energy consumption (saving up to 73.98% energy), minimized execution cost (up to 46.9%), and improved resource utilization (up to 31.8%). X. Chen et al. (2020) suggested a sophisticated strategy known as Improved WOA for Cloud task scheduling (IWC) to enhance the WOA-based method's capacity to find the best solution [69]. Simulation-based experiments demonstrated that the suggested IWC outperformed the existing metaheuristic algorithms in terms of convergence speed and accuracy when looking for the best task scheduling plans. A comprehensive implementation of IWC was also presented. Additionally, it performed better when using system resources when both little and large-scale workloads were present.

Table 4. Comparative Analysis of Metaheuristic based Task scheduling approaches

Author & Year	Technique Used	Dataset	Performance Metrics	Simulation Tool	Findings	Limitations
B. M. Hasani Zade et al. (2025)	Multi-Objective Improved Beluga Whale Optimization with Ring Topology (MO-IBWO-Ring)	HCSP (512 & 1024 tasks) + random	Makespan, cost, resource utilization	Custom + CloudSim	Outperformed DN-NSGAII, MO_Ring_PSO_SCD, Omni-optimizer, and MOPSO	Risk of local minima remains under very large-scale workloads
Jeng-	Advanc	CEC	Makes	Benc	Superior to	Not yet

Shyang Pan et al. (2025)	ed Willow Catkin Optimization (AWCO) with quasi-opposition & sinusoidal mapping	2014 benchmark suite	pan, cost, load balance	hmark tests	WCO and other metaheuristics; better convergence	validated on real cloud traces
Yan Han (2025)	PSO-based intelligent scheduling	Google Cloud dataset	Task completion time, energy consumption	Cloud Sim	Balanced user requirements & energy efficiency	May suffer from premature convergence in large workloads
Cebrail Barut et al. (2024)	Interpretable rule-based metaheuristic + ML two-phase (offline + online)	Random + archived datasets	Execution time, solution quality	Custom framework	Reduces scheduling time; reuses archived solutions effectively	Performance tied to size & quality of archive dataset
P. Tamilarasu et al. (2024)	Improved Coati Optimization Algorithm (ICOATS)	Google Cloud dataset	Makespan, resource utilization, turnaround efficiency	Cloud Sim	Better performance vs. metaheuristics; reduced makespan & cost	Exploration–exploitation tradeoff may still fail in extreme workloads
Laith Abualigah et al. (2024)	Improved Jaya + Synergistic Swarm Optimization + Levy flight	Google Cloud dataset	Accuracy, convergence speed, scalability	Cloud Sim	Achieved 88% accuracy; 10% better than original	Results rely on synthetic workloads
P. Pabitha et al. (2024)	Chameleon & Remora Search Optimization (CRSOA)	Google Cloud dataset	Makespan, cost, load balance, completion time	Cloud Sim	Reduced makespan (18.96%), cost (22.18%), imbalance (20.54%)	Computational overhead in multi-objective evaluation
Chirag Chandrashekar et al. (2024)	Modified Chimp-Whale Optimization Algorithm (MCWOA)	Structural models (beam, frame) + synthetic tasks	Energy, cost, delay, duration	Custom simulation	Outperformed COA, WOA, GA, ALO, GWO	Focused on structural detection; limited direct cloud benchmarks
Z. A. Khan et al.	Parallel Enhanced Whale	GoCJ + HCSP	Makespan, throughput,	Cloud Sim	Outperformed WOAmM, MRMPSO; scalable	Complexity of parallelization

(2024)	Optimization Algorithm (PEWOA)	datasets	resource utilization		with varying tasks	increases implementation cost
Z. Cui et al. (2023)	Multi-Objective Multi-Factorial Optimization (MO-MFO) with knowledge transfer	Cloud task test dataset	Execution time, cost, load balance	CloudSim	Improved efficiency & convergence vs. evolutionary algorithms	Similarity estimation for transfer may be costly
Sudheer Mangalampalli et al. (2023)	Multi-objective trust-aware scheduler using WOA	NASA + HPC2N workloads	Makespan, energy, trust (availability, success rate, turnaround)	CloudSim	Significant gains vs. ACO, GA, PSO	Trust parameters may not generalize beyond datasets
H. Zhang et al. (2023)	Improved Cat Swarm Optimization (CSO) with weight adjustment	Cloud test dataset + Ackley benchmark	Execution time, load, convergence	Custom	Better convergence (102 iterations; exec time 6.23s)	Risk of local optima without diversity strategies
Raja Masadeh et al. (2021)	Sea Lion Optimization Algorithm (SLnO)	Synthetic workloads	Makespan, cost, energy, imbalance	CloudSim	Saved 73.98% energy, 46.9% cost, improved utilization by 31.8%	Limited testing on large-scale workflows
X. Chen et al. (2020)	Improved Whale Optimization Algorithm (IWC)	Google Cloud dataset	Makespan, utilization, convergence speed	CloudSim	Outperformed existing WOA variants; better scalability	Needs validation on multi-objective settings

9.5 Hybrid Approaches for Cloud Task Scheduling

Mohaymen Selselejoo et al. (2025) presented a hybrid model for scheduling independent activities that combined the Grey Wolf Optimization (GWO) algorithm with the decision tree approach [70]. The model sought to maintain load balance, optimize makespan, lower overall costs, and improve resource use. The GWO algorithm assigned resources to the chosen tasks in the suggested method after the tasks were first categorized using a decision tree. The CloudSim toolkit was used to run simulations in a diverse setting. The experiments took into account a range of input possibilities, from 200 to 3200 tasks. The suggested DT-GWO hybrid model-maintained load balance while achieving improvements of at least 18.5% in makespan, 3.4% in average resource utilization, and 12.7% in overall cost when compared to the standalone GWO algorithm. N. Omran Alkaam et al. (2025) proposed an improved version of Henry Gas Solubility Optimization called Henry Gas-Harris Hawks-Comprehensive Opposition (HGHHC). Harris Hawks Optimization (HHO) and Comprehensive Opposition-Based Learning (COBL) served as the foundation for this approach [71]. This proposed technique improved the quality of permitted solutions by using the HHO algorithm as a local search strategy. COBL enhanced the less effective solutions by carefully examining their opposites and choosing an efficient one. This approach made it simpler to enhance inadequate solutions, which raised the selected strategies' overall efficacy. In terms of makespan (MKS), it performed 34.30, 72.95, and 28.67, in that order. The matching values for resource utilization (RU) were 16.92, 28.72, and 25.58. As a result, the

suggested HGHHC algorithm outperformed earlier methods in terms of simulated makespan and resource use. This demonstrated how well hybrid meta-heuristic algorithms balance exploration and exploitation, avoiding local optima.

Ipsita Behera et al. (2024) aimed to reduce search time while identifying a near-optimal solution for a multi-objective work scheduling problem in the cloud [72]. A common cloud computing task scheduling technique was proposed to enhance system performance and optimize the Quality of Service (QoS) parameters such as energy, makespan, resource utilization, and throughput by utilizing the advantages of Genetic Algorithms (GA) and Gravitational Search Algorithms (GSA) while avoiding their disadvantages. Standard functions, real-time, and synthetic workloads were all tested using CloudSim. Other comparable, metaheuristic-based methods that were assessed under the identical circumstances were contrasted with the results that were achieved. In terms of Degree of Imbalance (12%), resource utilization (9%), Mean Response Time (7%), and energy consumption (6%), the developed method performed better than GSA, ACO, and PS. Shasha Zhao et al. (2024) developed a hybrid of the Hierarchical Particle Swarm Optimization (HPSO) and Evolutionary Artificial Bee Colony (EABC) algorithms, known as the Hierarchical Particle Swarm Optimization-Evolutionary Artificial Bee Colony Algorithm (HPSO-EABC) [73]. The HPSO-EABC approach integrated the benefits of the HPSO and EABC algorithms. Thorough testing was done, which included assessments of load balancing, resource execution time, algorithm convergence speed, and operating expenses. According to the results, the EABC method was more parallel than the Artificial Bee Colony algorithm. As a result, the hybrid HPSO-EABC algorithm displayed considerable gains in terms of stability and convergence speed. Additionally, the ablation experiment confirmed that it demonstrated improved resource scheduling performance in both homogeneous and heterogeneous situations, successfully cutting execution time and cost.

B. Saemi et al. (2023) developed Hybrid Multi-objective Harris Hawks Optimization (HMHHO), a non-dominated multi-objective strategy based on the Harris Hawks Optimization (HHO) technique, to address the problem in MCC [74]. Allocating jobs from mobile source nodes to public cloud processors, cloud patches, and mobile resource processors were the goals of this study. The suggested approach, on average, finished tasks faster and consumed less energy than the other four algorithms: The Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Cuckoo Search Algorithm (CSA). B. Kruekaew et al. (2022) suggested a self-contained method for cloud computing task scheduling that combines a Multi-Objective Task Scheduling Optimization based on the Artificial Bee Colony Algorithm (ABC) with a Q-learning algorithm, a reinforcement learning technique that accelerated the ABC algorithm [75]. In order to overcome the limits of concurrent considerations, the suggested solution sought to optimize scheduling and resource utilization, maximize virtual machine throughput, and provide load balancing between VMs based on makespan, cost, and resource utilization. Using three datasets—Random, Google Cloud Jobs (GoCJ), and Synthetic workload—performance study of the suggested approach was conducted using CloudSim in comparison to the current load balancing and scheduling methods: Max-Min, FCFS, HABC_LJF, Q-learning, MOPSO, and MOCS algorithms. According to the experimental data, the algorithms that employed the MOABCQ technique performed better than the others in terms of throughput, average resource consumption, makespan reduction, cost reduction, and degree of imbalance reduction.

K. Li et al. (2022) suggested a method for scheduling tasks in cloud computing that combines Membrane Computing and Particle Swarm Optimization [76]. The first step was to propose a task scheduling model with time and cost functions as the target. Then, using the Particle Swarm algorithm, the adaptive weight factor based on a sinusoidal function was used to prevent the algorithm from reaching a local optimum. Finally, chaos operation was used in population initialization to enhance the diversity of rich understanding. Finally, the performance of the PSOMC method was demonstrated by comparing six benchmark test functions in simulation tests. To improve the caliber of individual solutions, Membrane Computing was applied to individual screening. Additionally, it was confirmed that, for varying numbers of tasks, the consumption cost and completion time were noticeably superior to those of the ACO, PSO, and MC algorithms. N. Manikandan et al. (2022) suggested a solution using fuzzy C-Means clustering hybrid algorithms that use Fish Swarm Optimization for effective resource allocation and Black Widow Optimization for job scheduling in order to lower costs, energy consumption, and resource usage [77]. Fuzzy C-Means was used to cluster the virtual machines, the Fish Swarm Algorithm was used to schedule tasks, and Black Widow Optimization was used to optimize resource allocation. Three other algorithms' performances were compared to the suggested solution's, exposing certain shortcomings in the current methods. In terms of lower makespan, better load balancing, and increased resource utilization efficiency, the suggested approach produced better outcomes.

S. Velliangiri et al. (2021) suggested Hybrid Electro Search with a Genetic Algorithm (HESGA) to enhance work scheduling behavior by taking into account variables including makespan, load balancing, resource usage, and multi-cloud cost [78]. The advantages of an electrosearch algorithm and a genetic algorithm were integrated in the suggested strategy. The best global optimal solutions were produced by the Electro Search Algorithm, while the best local optimal solutions were produced by the Genetic Algorithm. The suggested approach performed better than other scheduling algorithms, including GA, ES, ACO, and Hybrid Particle Swarm Optimization Genetic approach (HPSOGA). M.S. Sanaj et al. (2021) suggested a productive method for job scheduling in the specified cloud that makes use of GA-WOA and the MAP Reducing architecture [79]. First, the client's task was used to extract the task features. The MRQFLDA algorithm was then used to decrease the features. A Map-Reduce structure was then used to divide the big tasks into smaller ones. Lastly, the GA-WOA algorithm was used to efficiently schedule the work. The CloudSim environment was used to run the experimental simulations. The outcomes demonstrated that, across a range of evaluation indicators, the suggested GA-WOA approach performed better than the other approaches. M. Abd Elaziz et al. (2019) offered a fresh approach to the cloud task scheduling problem that sought to reduce the amount of time needed to plan many tasks across various Virtual Machines (VMs). The suggested approach was founded on the use of Differential Evolution (DE) to enhance the Moth Search Algorithm (MSA) [80]. The MSA used two concepts—the phototaxis and Levy flights, which stood for the ability to explore and exploit, respectively—to mimic how moths would fly toward the source of light in the natural world. Three experimental series were conducted in order to assess the suggested MSDE algorithm's performance. The performance of the suggested approach to resolve the cloud task scheduling problem was contrasted with alternative heuristic and meta-heuristic algorithms using synthetical and real trace data, respectively, in the second and third experimental series. The suggested algorithm performed better than other algorithms based on performance metrics, as demonstrated by the outcomes of the two-testing series. S. Pang et al. (2019) developed an estimation of distribution algorithm (EDA-GA) hybrid scheduling method that blends the ideas of genetic algorithms [81]. First, a specific scale of workable solutions was produced using the EDA sampling approach and probability model. Second, the search range of solutions was increased by utilizing GA's crossover and mutation processes. At last, the best scheduling method for allocating jobs to virtual machines was identified. Strong search capabilities and quick convergence were two benefits of this method. The algorithm proposed in this paper was compared with EDA and GA using the CloudSim simulation experiment platform. According to the experimental findings, the EDA-GA hybrid algorithm may successfully shorten task completion times while enhancing load balancing capabilities.

Table 5. Comparative Analysis of Hybrid approaches for Task scheduling

Author & Year	Technique Used	Dataset	Performance Metrics	Simulation Tool	Findings	Limitations
Mohaymen Selselejoo et al. (2025)	Hybrid Decision Tree + Grey Wolf Optimization (DT-GWO)	Heterogeneous Cloud, 200–3200 tasks	Makespan, cost, resource utilization, load balance	CloudSim	Improved makespan (18.5%), resource utilization (3.4%), cost (12.7%) vs. GWO	Performance compared only to baseline GWO
N. Omran Alkaam et al. (2025)	Hybrid Henry Gas Solubility + HHO + COBL (HGHHC)	NASA, HPC2N, Synthetic datasets	Makespan, resource utilization	CloudSim	Better MKS (34.30, 72.95, 28.67) and RU (16.92, 28.72, 25.58)	Complexity of hybridization may increase runtime
Ipsita Behera et al. (2024)	Hybrid GA + GSA	Standard + Real-time + Synthetic workloads	QoS (energy, makespan, utilization, throughput), Mean Response	CloudSim	Outperformed GSA, ACO, PSO by 6–12% on key metrics	Scalability on very large workloads not tested

			Time, imbalance			
Shasha Zhao et al. (2024)	Hybrid Hierarchical PSO + Evolutionary ABC (HPSO-EABC)	Synthetic workloads	Convergence speed, execution time, load balancing, cost	Custom + Ablation tests	Improved stability, convergence, reduced execution time & cost	Complexity of hybrid structure may hinder adaptability
B. Saemi et al. (2023)	Hybrid Multi-Objective Harris Hawks Optimization (HMHHO)	MCC workloads	Makespan, energy	Custom cloud-MCC	Outperformed GA, ACO, PSO, CSA	Evaluated only for mobile-cloud offloading
B. Kruekaew et al. (2022)	MOABC + Q-learning (MOABCQ)	Random, Google Cloud Jobs (GoCJ), Synthetic workloads	Makespan, cost, resource utilization, throughput, load balance	CloudSim	Superior to Max-Min, FCFS, MOPSO, etc.; improved makespan & RU	Depends on effective Q-learning parameter tuning
K. Li et al. (2022)	Hybrid PSO + Membrane Computing (PSOMC)	Benchmark + cloud workloads	Completion time, cost	CloudSim	Better results than ACO, PSO, MC; chaos initialization improved diversity	Parameter tuning increases complexity
N. Manikandan et al. (2022)	Fuzzy C-means + Fish Swarm + Black Widow Optimization	Synthetic workloads	Makespan, cost, energy, resource utilization	CloudSim	Reduced makespan, improved load balance & RU	Complexity in combining clustering + hybrid optimization
S. Velliangiri et al. (2021)	Hybrid Electro Search + GA (HESGA)	Multi-cloud workloads	Makespan, load balance, utilization, cost	Matlab	Outperformed HPSOGA, GA, ES, ACO	Validation limited to multi-cloud
M.S. Sanaj et al. (2021)	GA + Whale Optimization + MapReduce framework	Real & synthetic workloads	Makespan, scheduling efficiency	CloudSim	Outperformed baseline methods; efficient handling of large tasks via MapReduce	Extra preprocessing cost due to feature extraction
M. Abd Elaziz et al. (2019)	Improved Moth Search Algorithm + Differential	Synthetic + real trace data	Makespan	Benchmark + CloudSim	Outperformed MSA, heuristics, metaheuristics	Focused mainly on makespan; ignores multi-objective

	Evolution (MSDE)					
S. Pang et al. (2019)	Hybrid Estimation of Distribution Algorithm + GA (EDA-GA)	CloudSim workloads	Task completion time, load balancing	CloudSim	Fast convergence, reduced completion time, better load balance	Lacks cost and energy efficiency analysis

10. Objectives Of Task Scheduling

The primary goal of task scheduling is to match the anticipated objectives by assigning tasks to the appropriate nodes for execution [17]. In dynamic cloud computing environments, the main objectives include a wide range of topics such as load balancing, economic principles, reducing execution times, optimization resource consumption rates, and improving adaptability.

- Economic Principle: The resource nodes in cloud/grid computing systems are frequently dispersed among several areas, and consumers are required to pay the resource provider reasonable administration fees. Economic considerations must be made when scheduling tasks in order to satisfy the needs of both suppliers and consumers and ensure that the network's resource nodes continue to grow sustainably.
- Minimum Execution Time: The primary objective of task scheduling is to decrease the overall execution time. This time span, which spans from the start of the first task to the completion of the last task, is significantly influenced by task durations and intertask communication costs.
- Efforts are concentrated on minimizing the influence of these components in order to decrease the overall execution time.
- Maximum Resource Utilization: The effectiveness of how the scheduling algorithm uses the resources that are available is reflected in the resource utilization efficiency across the network's execution nodes. When comparing systems with and without load balancing methods, the former usually show greater resource consumption efficiency [18].
- Load Balancing: Task scheduling methods for Cloud/Grid scenarios must be designed with efficient load balancing between resource nodes. Without appropriate load balancing techniques, resources could be wasted as certain nodes might be overburdened while others are left unused.
- Dynamic Adaptability: Nodes in cloud and grid computing systems have the ability to join and exit the network at any time. To preserve optimal performance in the face of network state diversification, the scheduling algorithm must take this dynamic nature into account.
- Uncertainty Control: In cloud environments, uncertainty management is a major concern. It is possible to greatly improve scheduling accuracy by creating techniques for quantifying and controlling uncertainty. As an illustration, scheduling decisions may be impacted by the fluctuating performance of virtual machines during execution, underscoring the significance of skillfully managing uncertainties to guarantee task execution success.

11. Overview Of Scheduling Constraints In Cloud Computing

In cloud scheduling, constraints like deadlines, priorities, budgets, and fault tolerance are essential, as shown in Figure 4. These factors can harm the Service Level Agreement (SLA) when many applications fail to meet them.

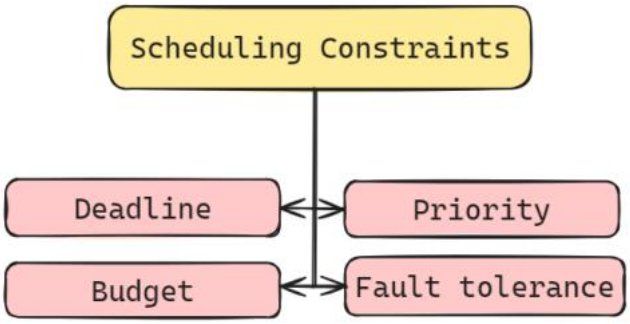

Fig.4. Scheduling constraints classification

The following section explores selected mechanisms that address these constraints and their main features.

- Budget: The budget is a limit set by the user on the use of the cloud service provider's resources. Scheduling decisions consider this budget limit to reduce the overall execution time of the workflow, task, or job and ensure it is completed within the budget. In practice, budget-constrained scheduling aims to balance cost and performance. Resource allocation should prioritize low-cost VMs while avoiding unnecessary delays. Effective budget-aware scheduling helps users make the most of their resources without going over their financial limits.
- Deadline: Applications that depend on timing must finish their work within a set timeframe. These applications use deadline-constrained scheduling to ensure results are delivered on time. This type of scheduling must also consider associated costs. For time-sensitive applications, strong deadline-constrained scheduling is vital as it boosts the program's reliability. Moreover, missing deadlines may lead to penalties, SLA violations, or even system instability, making deadline-aware mechanisms essential in mission-critical workflows [19].
- Fault Tolerance: Fault tolerance is vital for achieving high performance in cloud computing. It ensures that tasks continue running even with hardware or software failures. Methods such as task replication, checkpointing, and resubmission are often used to provide fault tolerance in scheduling strategies. A fault-tolerant scheduler improves system availability, reduces the risk of SLA violations, and ensures reliable service in unpredictable cloud environments. Therefore, it is essential for applications that require ongoing, uninterrupted processing.
- Priority: Priority-based scheduling ensures that important or urgent tasks are executed before less critical ones. This method is especially helpful when resources are scarce and multiple tasks compete for execution. Prioritization typically depends on business needs, user demands, or application requirements. A well-designed priority scheduling system can significantly boost user satisfaction by addressing critical tasks first. However, it is important to prevent resource starvation for lower-priority tasks, which can be managed with hybrid or fairness-aware scheduling models [20].

12. Testing Tools

Testing tools are essential in many domains, such as cloud computing and assembly procedures, because they offer platforms for assessing and improving efficiency, cost, and performance. Simulation tools are essential for evaluating novel scheduling algorithms in cloud computing and evaluating how well they function in different circumstances without sacrificing end-user Quality of Service (QoS). As shown in Table 6, some of the most well-known simulation programs include GridSim, CloudSim, Matlab, CloudAnalyst, WorkflowSim, GridSim, and CloudSim Plus. Table 6 of this study makes it abundantly evident that CloudSim is frequently used for task scheduling because of its expandable programmatic classes, despite its poor graphical assistance and modelling of energy use. Furthermore, as numerous studies have shown, the best results from several comparing algorithms can be used to compare the efficacy of various algorithms. In order to ensure that researchers and practitioners can successfully implement and test novel methodologies, the selection and comparison of testing tools ultimately depend on certain needs like time efficiency, applicability, flexibility, and the level of support offered by the simulator.

Table 6. Summary of the testing tools

Testing Tool	References
CloudSim	[20], [21], [22], [25], [26], [27], [30], [31], [32], [33], [34], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50], [51], [59], [60], [61], [62], [63], [64], [65], [71], [72]
CloudSim Plus	[34]
CloudSim with CloudSim Plus	[45]
MATLAB with CloudSim	[77]
MATLAB with CloudSim	[44], [41], [45]
Not Specified	[24], [25], [28]
Python Toolbox	[33], [35], [38], [42]
WorkflowSim Framework	[25],[37]

13. Datasets

It is essential to many areas of study since it gives the data needed to evaluate, model, and validate diverse approaches. In order to capture natural workload behaviour, for example, the Google Cloud dataset [35], [55], [66] which comprises MapReduce logs [57], [72] and Google cluster traces, is used. Tasks are grouped according to job size in terms of Million Instructions (MI). Aspects of cloud computing environments, including consumption patterns and virtual machine lifespan, have also been studied using alternative datasets, such as the Microsoft Azure VM resource usage traces and the Alibaba cluster traces. Datasets are crucial for assessing how well detection methods work in the field of intrusion detection. These datasets, which are used to discover important aspects for comparing other datasets and evaluating intrusion scenarios, usually comprise network traffic information, either packet-based or flow-based. IoT network traffic data is available for these kinds of analysis via a number of sources, such as AZSecure, Contagiodump, and Kaggle. Because they offer the fundamental facts needed for thorough analysis and validation, datasets are essential for furthering research in a variety of fields.

14. Research Techniques

The overview of our thorough and organised analytical method for assessing task scheduling in cloud computing is given in this part. We arrange our results based on a variety of factors, including methodologies, strategies, datasets, and evaluation measures, by using particular search criteria. The four-phase process used to perform this systematic literature review (SLR) includes developing research questions, carrying out search operations, choosing pertinent papers, and synthesizing data. The objective is to conduct a methodical analysis of the state of cloud computing task scheduling. Future research efforts in this area will benefit from this evaluation approach, which makes it easier to identify research gaps and improves our comprehension of the basic ideas and factors related to task scheduling in cloud computing. The following highlights this systematic literature review's main contributions:

- The methods, issues, metrics, datasets, algorithms, and approaches in metaheuristic task scheduling are all covered in this work's qualitative and quantitative review of the state-of-the-art task scheduling in cloud computing.
- In order to help researchers and general practitioners in the field better understand the emerging trends and identify solutions for cloud task scheduling problems, this work presents a systematic literature review of the current task scheduling in cloud computing models.
- This article identifies the future roadmap for task scheduling in cloud computing and provides thorough descriptions of open topics and obstacles.

14.1 Research Questions

Primary research and the creation of a new framework to address some of the unresolved concerns listed in the study can benefit from the responses to the following questions, which are specific to metaheuristic-based systems. The prepared review questions are shown in Table 7, along with the justification for the research questions.

Table 7. Research questions (RQ).

RQ No.	Research Question	Motivation
RQ1	What types of task scheduling techniques have been applied in cloud computing in prior studies?	To review existing work in the field and gain insights into the most widely explored as well as emerging scheduling approaches.
RQ2	What evaluation metrics are commonly employed to assess the performance of task scheduling techniques in cloud computing?	To identify the standard parameters used for validating and comparing scheduling methods, ensuring a consistent basis for performance analysis.

14.2 Search Strategy

An automated search was performed on the major digital libraries, including IEEE Explores, ACM, Springer, ScienceDirect, Elsevier, Taylor & Francis, and MDPI, in order to collect the papers that are pertinent to this systematic review. Because of their acceptance, large collection of research articles, and abundance of research articles pertaining to our study topics, the libraries were chosen. The number of selected papers on each digital library is shown in Figure 5, and the number of publications annually from 2018 to June 2025 is shown in Figure 6. The following keyword combinations were used to narrow down the search's scope: "(("task scheduling" OR "resource scheduling"); AND ("meta-heuristic optimisation" OR "meta-heuristics") AND ("cloud" OR "cloud computing"))"; AND ("Machine Learning") AND ("Hybrid Approaches") AND ("Heuristics OR Heuristic based Approaches") AND ("Traditional Approaches OR "Algorithms").

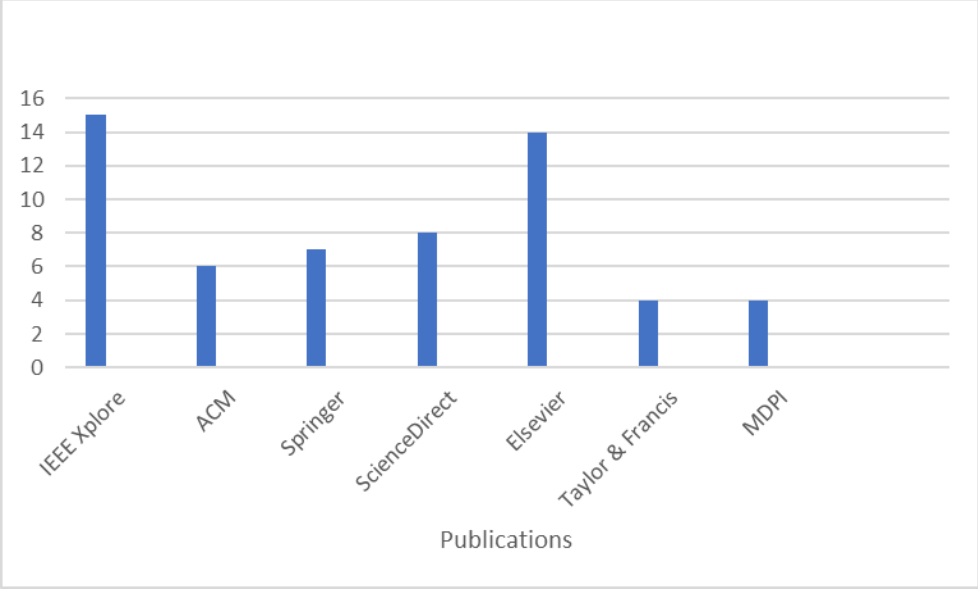

Fig.5. Publication platforms

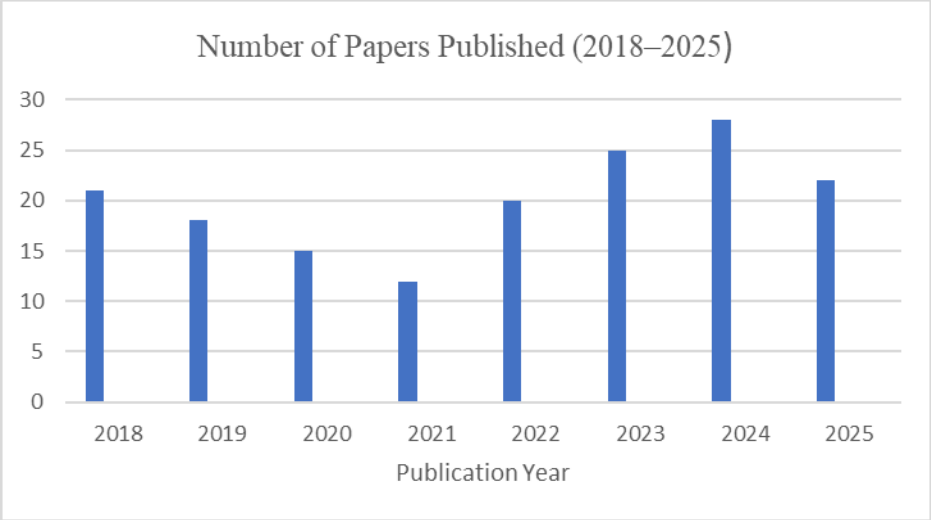

Fig.6. Number of publications per year

14.3 Screening and Selection Processes

The method of choosing papers to find the most pertinent research, which follows the search strategy, is known as the paper selection criterion. Table 8 lists the inclusion and exclusion criteria for the 58 relevant papers on cloud computing job scheduling that were chosen.

Table 9. Inclusion and exclusion criteria.

Inclusion Criteria	Exclusion Criteria
Papers published between January 2018 and June 2025 only.	Papers published before 2018.
At least one meta-heuristic technique is applied in the selected study.	Studies that do not consider any meta-heuristic technique.
Task scheduling in cloud computing is the primary focus of the study.	Studies where other resource management issues are addressed but task scheduling is not fully considered.
Only peer-reviewed journal articles and conference papers were included (with preference for Elsevier and IEEE publications).	Non-peer-reviewed articles, magazines, or blogs.
Papers written in the English language.	Papers written in languages other than English.
Research articles that address at least one of the research questions (RQ1 or RQ2).	Papers not related to RQ1 or RQ2.
The study must relate specifically to task scheduling in cloud computing.	Studies not published or incomplete.
Priority is given to articles published in Elsevier and IEEE journals/conferences.	Duplicate papers.
Only published research papers (journals, conferences, symposium proceedings).	PhD theses, Master's dissertations, and books are excluded.

The selected papers, all in English focused on at least one chosen research question. Specifically, the study encompassed articles published between 2018 and June 2025, including both conference and journal publications. The paper selection process adhered to the PRISMA guidelines. The papers were selected from nine selected

international publications and were analyzed against research questions (RQ). Each paper's abstract and methodology sections were carefully examined, and the search results were filtered based on predetermined criteria. The selection process comprised four stages: identification, screening, eligibility, and inclusion. Initially, an automated search yielded 1970 articles. Following the screening process, 58 articles remained after removing duplicates and inaccessible sources.

15. Results And Discussion

Traditional approaches in cloud task scheduling mainly focus on improving resource utilization, reducing makespan, and optimizing response time. For instance, Reddy et al. (2025) and Pachipala et al. (2024) enhanced traditional scheduling algorithms like RAS and SJF to tackle resource bottlenecks and job prioritization effectively. Similarly, Sanaj et al. (2020) and Choudhary et al. (2022) refined Round Robin-based methods to minimize waiting time and task migration, while Aref et al. (2022) proposed a hybrid strategy combining Min-Min, Max-Min, and GA for better load balancing. Overall, these studies show that while traditional algorithms remain foundational, their limitations in scalability and adaptability drive the need for more advanced or hybrid solutions. Machine learning, particularly deep reinforcement learning (DRL), has emerged as a powerful approach for cloud task scheduling, addressing challenges like load balancing, energy efficiency, and workflow optimization. Recent works (J. Wang et al., 2025; J. Anand et al., 2025) demonstrate the adaptability of RL-based methods through dynamic reward mechanisms and real-time load awareness, while others (H. Hou et al., 2024; Xinglong Pei et al., 2024) focus on reducing response time and energy consumption. Multi-agent frameworks (D. Changzhi et al., 2024; A. Jayanetti et al., 2024) further enhance adaptability and sustainability in distributed environments. However, despite these advancements, challenges remain in scalability, generalization, and convergence stability, leaving room for further research (X. Wang et al., 2023; L. Ran et al., 2019). Heuristic-based scheduling approaches in cloud computing emphasize efficiency, fault tolerance, and energy optimization. Techniques such as replication and migration improve fault tolerance and resource utilization (S. Chawla et al., 2024), while algorithms like CACS optimize workflow scheduling under specific system restrictions (W. Chongdarakul et al., 2024) [2]. Multi-objective models like TSO-MCR further balance trade-offs among makespan, cost, and reliability (Boroumandp et al., 2024). Additionally, energy-aware heuristics such as ECOTS significantly reduce power consumption without compromising performance (Weiwei Lin et al., 2018). Overall, heuristic strategies provide lightweight yet effective solutions for diverse scheduling objectives in dynamic cloud environments. Recent research on cloud task scheduling highlights the rise of advanced metaheuristic and hybrid optimization methods. For example, Hasani Zade et al. (2025) introduced the MO-IBWO-Ring algorithm to overcome local minima stagnation, while Pan et al. (2025) proposed AWCO with quasi-opposition learning to boost convergence. Similarly, Barut et al. (2024) combined machine learning with metaheuristics for interpretable scheduling, whereas Tamilarasu et al. (2024) and Pabitha et al. (2024) focused on reducing makespan and improving load balancing through improved coati and hybrid chameleon-remora strategies. Despite these advances, studies such as Khan et al. (2024) and Cui et al. (2023) emphasize that scalability, energy efficiency, and robustness across dynamic environments remain ongoing challenges. Hybrid approaches for cloud task scheduling have gained significant attention due to their ability to balance exploration and exploitation while optimizing multiple objectives. For example, Selselejoo et al. (2025) combined decision trees with Grey Wolf Optimization to improve makespan and cost efficiency, while Alkaam et al. (2025) enhanced Henry Gas Optimization with HHO for better scalability. Similarly, Behera et al. (2024) and Zhao et al. (2024) demonstrated that combining GA with GSA, or PSO with ABC variants, leads to superior performance in resource utilization and convergence speed. Earlier studies, such as Velliangiri et al. (2021) and Pang et al. (2019), also confirmed that hybridizing classical heuristics with metaheuristics provides more robust scheduling strategies. Collectively, these works highlight that hybrid models outperform standalone methods, though challenges remain in scalability and adaptability across diverse cloud environments. The contribution of different methods in the field of scheduling algorithms is illustrated in Figure 7, especially for cloud computing environments. Research in this area usually consists of separate, multi-objective investigations. The main goal of the majority of cloud task scheduling algorithms is to reduce makespan, or the overall amount of time needed to finish a group of activities, as shown in Figure 8. Other scheduling goals, such as cost and energy efficiency, are also deemed significant, though. The literature pays less attention to security and dependability characteristics, despite their importance.

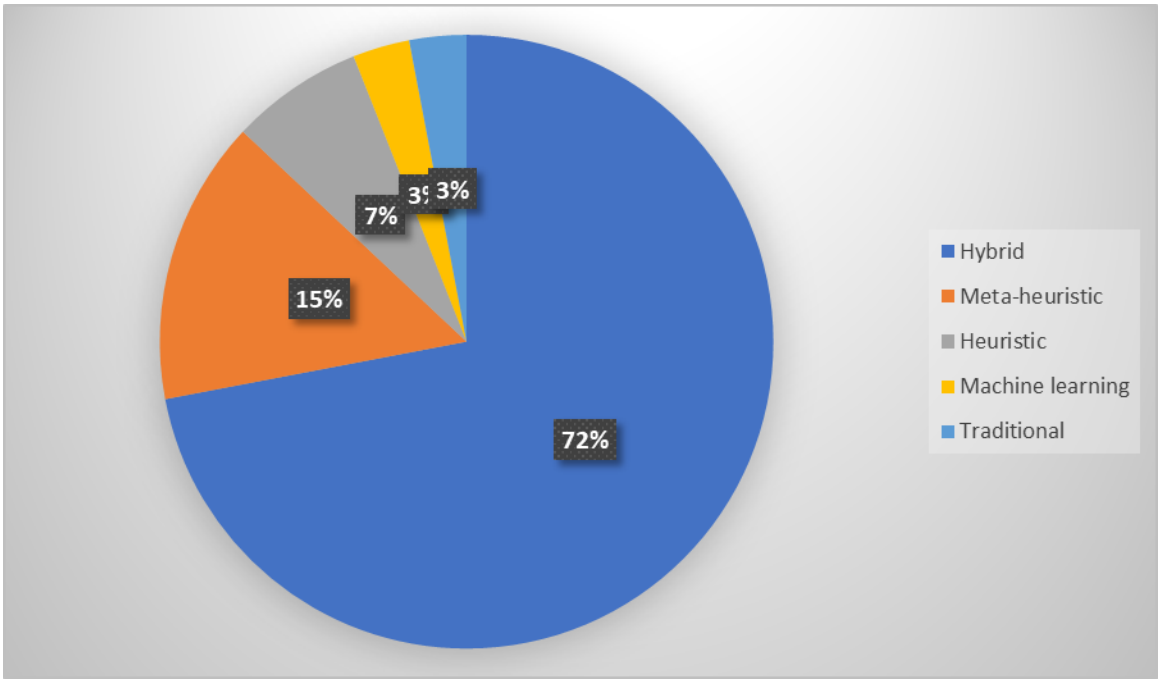

Fig. 7. Percentage distribution of Task scheduling techniques.

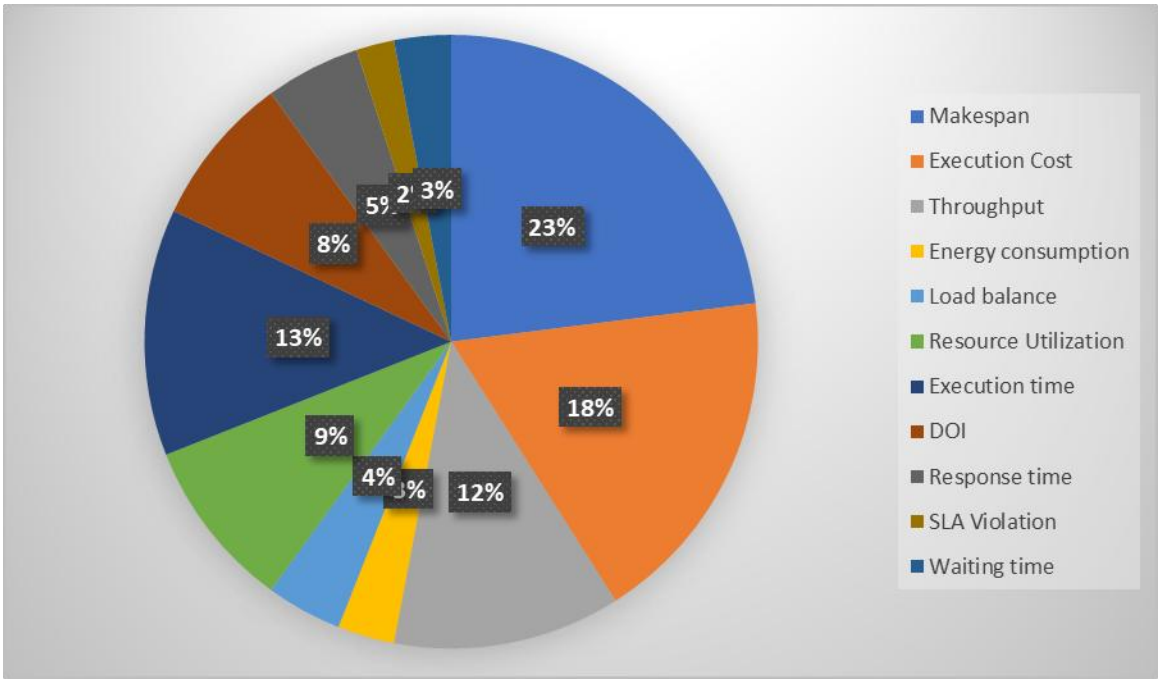

Fig. 8. Percentage distribution of scheduling objectives.

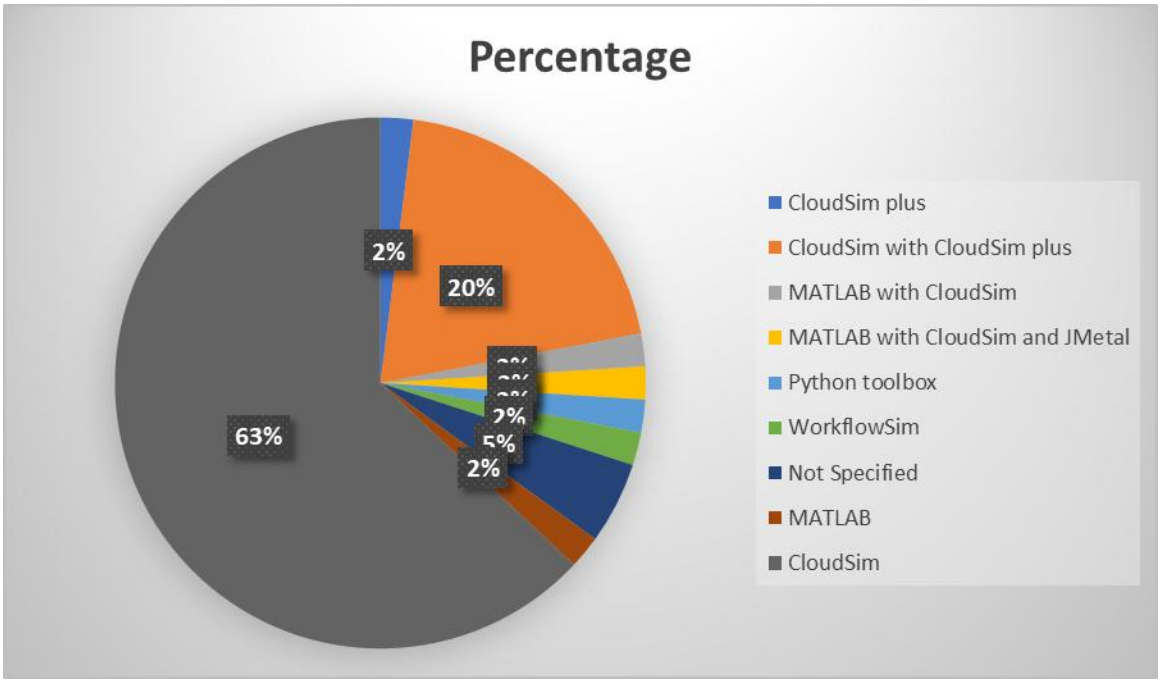

Fig. 8. Percentage distribution of task scheduling testing tools

As seen in Figure 8, the CloudSim simulation toolbox is frequently used for simulating and assessing various scheduling strategies. Researchers can modify CloudSim's pre-existing programmatic classes to suit their algorithms' needs, making it easier to evaluate different Quality of Service (QoS) metrics. Researchers may efficiently create and assess scheduling algorithms that satisfy a variety of goals and limitations in cloud computing settings by utilising these tools and methodologies.

16. Limitations

Some limitations of this work are:

- The papers in this review are based only on English-language publications; non-English publications are excluded.
- The study looks at various approaches, including traditional methods, machine learning, heuristic methods, metaheuristic methods, and hybrid methods. However, the evaluation is limited to specific datasets and performance metrics. Therefore, the findings may not apply to different cloud environments or real-world large-scale workloads.
- The study does not cover all constraints and QoS parameters, such as load balancing, storage, and bandwidth. In addition, it does not adequately explore aspects like realistic applications.
- This work references some papers that need to be purchased; they may be relevant to this study.
- The selection of papers is based on criteria like the title, abstract, keywords, and in some cases, the full text. This approach may have missed some relevant articles.
- This work only focuses on peer-reviewed journal articles and conference papers; we might have overlooked some articles in other formats.

17. Future Research Issues

Pay-per-use cloud computing has advanced the implementation of reliable computer infrastructures significantly. Even though several algorithm types have been effectively used in a wide range of situations, there is still a lot of room for additional research and discussion on a number of current issues and subjects. To improve scheduling efficiency, future studies should concentrate on successfully combining task scheduling with virtual machine consolidation techniques. Achieving the shortest work duration (maximising system throughput) while preserving the required degree of quality of service (QoS) should be the aim of efficiently using the fewest resources possible. No single algorithm can solve every problem, as demonstrated by the different cutting-edge scheduling

techniques examined in this paper. For instance, one algorithm may prioritise resource utilisation, availability, response speed, and scalability, but it may ignore other algorithms' priorities for energy efficiency, cost, time, and quality of service. Energy efficiency, load balancing, and scalable virtual machine migration are just a few of the scheduling goals that can be creatively optimised using hybrid algorithms, either separately or in combination. To overcome these obstacles, further research and development is needed to improve current algorithms and provide fresh approaches that can adapt to the changing needs of cloud computing systems.

18. Conclusion

Cloud computing is a user-focused approach that offers on-demand access to various virtual resources. This access allows users to effectively use computing power, storage, and services for different tasks. Among the many elements of cloud computing, task scheduling is key for making the best use of resources, cutting execution time, and improving overall system performance. Good task scheduling affects user satisfaction, costs, and the scalability of cloud services. This makes it an important area for ongoing research. This systematic literature review looks at 58 research articles published from 2018 to June 2025, specifically addressing scheduling techniques in cloud computing. The articles are organized and examined based on their methods, the factors they consider, and their reported challenges. The review highlights the increasing importance of creating smart and flexible scheduling systems that can keep up with the rising complexity of cloud workloads and user demands. The findings show that while there has been significant progress using traditional, machine learning, heuristic, metaheuristic, and hybrid methods, several challenges remain. In particular, improving the availability, reliability, and strength of task scheduling is still a problem, especially in fast-changing and mixed environments. Additionally, cloud systems need to handle uncertainties like changing workloads, unexpected failures, and energy efficiency issues, which current methods do not completely address. Researchers have often faced constraints due to the tools, techniques, and computing setups available during their studies. As a result, many proposed methods show good results in controlled settings but encounter limitations when applied to large, real-world situations. The review points out that even though many techniques consider costs, execution time, energy use, and load balancing, the vast scope of cloud computing makes it difficult to optimize all goals at once. Another key point is that many algorithms have adjustable structures, meaning they can be modified or combined with others to improve performance. This creates significant potential for future research, especially in developing systems that can handle multiple goals, adapt, and learn on their own. With the rising integration of artificial intelligence, edge computing, and Internet of Things (IoT) technologies into cloud services, there is growing demand for scheduling strategies that are scalable, secure, and aware of their context. Moreover, areas like energy efficiency, data security, fault tolerance, and compliance with service-level agreements (SLAs) are still not thoroughly explored in many studies. Future developments in these matters could not only boost scheduling performance but also improve the sustainability and reliability of cloud services. This review concludes that while considerable progress has been made, task scheduling in cloud computing remains a complex challenge, offering many opportunities for innovation and improvement.

References

1. M. Sardaraz and M. Tahir, "A Hybrid Algorithm for Scheduling Scientific Workflows in Cloud Computing," in IEEE Access, vol. 7, pp. 186137-186146, 201
2. Y. -H. Jia et al., "An Intelligent Cloud Workflow Scheduling System with Time Estimation and Adaptive Ant Colony Optimization," in IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 51, no. 1, pp. 634-649, Jan. 2021
3. H. Ali, M. S. Qureshi, M. B. Qureshi, A. A. Khan, M. Zakarya and M. Fayaz, "An Energy and Performance Aware Scheduler for Real-Time Tasks in Cloud Datacentres," in IEEE Access, vol. 8, pp. 161288-161303, 2020
4. M. Soualhia, F. Khomh and S. Tahar, "A Dynamic and Failure-Aware Task Scheduling Framework for Hadoop," in IEEE Transactions on Cloud Computing, vol. 8, no. 2, pp. 553-569, 1 April-June 2020
5. G. Fan, L. Chen, H. Yu and D. Liu, "Modeling and Analyzing Dynamic Fault-Tolerant Strategy for Deadline Constrained Task Scheduling in Cloud Computing," in IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 50, no. 4, pp. 1260-1274, April 2020
6. H. R. Faragardi, M. R. Saleh Sedghpour, S. Fazliahmadi, T. Fahringer and N. Rasouli, "GRP-HEFT: A Budget-Constrained Resource Provisioning Scheme for Workflow Scheduling in IaaS Clouds," in IEEE Transactions on Parallel and Distributed Systems, vol. 31, no. 6, pp. 1239-1254, 1 June 2020
7. K. Pradeep and T. P. Jacob, "Comparative analysis of scheduling and load balancing algorithms in cloud environment," 2016 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2016, pp. 526-531
8. S. Gao, Y. Li and H. Huang, "A Multi-Objective Task Scheduling Method based on ACO in Cloud Environment," 2020 IEEE 6th International Conference on Computer and Communications (ICCC), 2020, pp. 1554-1559

9. N. Chaudhary and M. Kalra, "An improved Harmony Search algorithm with group technology model for scheduling workflows in cloud environment," 2017 4th IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics (UPCON), 2017, pp. 73-77
10. L. Kapoor et al., "SLOPE: A Self Learning Optimization and Prediction Ensembler for Task Scheduling," 2018 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), 2018, pp. 1-7
11. T. Monisha, M. Mekala, K. Pradeep, N. Gobalakrishnan and L. J. Al, "A Study on QoS based Task Scheduling using Meta Heuristic Algorithms in Cloud Environment," 2019 International Conference on Intelligent Computing and Control Systems (ICCS), 2019, pp. 653-657
12. H. Chen, X. Zhu, D. Qiu and L. Liu, "Uncertainty-Aware Real-Time Workflow Scheduling in the Cloud," 2016 IEEE 9th International Conference on Cloud Computing (CLOUD), 2016, pp. 577-584
13. S. H. Adil, K. Raza, U. Ahmed, S. S. A. Ali and M. Hashmani, "Cloud task scheduling using nature inspired meta-heuristic algorithm," 2015 International Conference on Open-Source Systems & Technologies (ICOSST), 2015, pp. 158-164
14. 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," in IEEE Access, vol. 10, pp. 17803-17818, 2022
15. X. Chen, K. Zhang, Z. Li and H. Wang, "The Application of Genetic Algorithm in Task Scheduling," 2015 International Conference on Network and Information Systems for Computers, 2015, pp. 332-334
16. Y. Yan, K. Du, L. Wang, H. Niu and X. Wen, "MP-DQN Based Task Scheduling for RAN QoS Fluctuation Minimizing in Public Clouds," 2022 IEEE International Conference on Communications Workshops (ICC Workshops), 2022, pp. 878-884
17. K. X. Kang, D. Ding, H. M. Xie, Q. Yin and J. Zeng, "Adaptive DRL-based Task Scheduling for Energy-Efficient Cloud Computing," in IEEE Transactions on Network and Service Management
18. M. Adhikari and T. Amgoth, "Efficient algorithm for workflow scheduling in cloud computing environment," 2016 Ninth International Conference on Contemporary Computing (IC3), 2016, pp. 1-6
19. V. A. Lepakshi and C. S. R. Prashanth, "Efficient Resource Allocation with Score for Reliable Task Scheduling in Cloud Computing Systems," 2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), 2020, pp. 6-12
20. D. Jain and A. Goutam, "Optimization of resource and task scheduling in cloud using random forest," 2017 International Conference on Advances in Computing, Communication and Control (ICAC3), 2017, pp. 1-5
21. K. J. Reddy, P. Udayaraju, V. P. R. Pamaiahgari and V. D. Kumar, "Implementing Resource-Aware Scheduling Algorithm for Improving Cost Optimization in Cloud Computing," 2025 4th International Conference on Sentiment Analysis and Deep Learning (ICSADL), Bhimdatta, Nepal, 2025, pp. 282-287, doi: 10.1109/ICSADL65848.2025.10933319.
22. Y. Pachipala, Kavya Sri Sureddy, A.B.S. Sriya Kaitepalli, Nagalakshmi Pagadala, Sai Satwik Nalabothu, Mihir Iniganti, “Optimizing Task Scheduling in Cloud Computing: An Enhanced Shortest Job First Algorithm,” Procedia Computer Science, Volume 233, 2024, Pages 604-613, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2024.03.250.
23. M. Prashanthi and B. Lalitha, "Analyzing Dynamic Workload Management and Energy-Efficient Task Scheduling in Cloud Computing," 2024 9th International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 2024, pp. 1035-1042, doi: 10.1109/ICCES63552.2024.10860200.
24. I.S. Aref, J. Kadum and A. Kadum, "Optimization of Max-Min and Min-Min Task Scheduling Algorithms Using G.A in Cloud Computing," 2022 5th International Conference on Engineering Technology and its Applications (IICETA), Al-Najaf, Iraq, 2022, pp. 238-242, doi: 10.1109/IICETA54559.2022.9888542.
25. Y. Bo, L. Feng and Z. Xiaoyu, "Cloud computing task scheduling algorithm based on dynamic priority," 2022 IEEE 6th Information Technology and Mechatronics Engineering Conference (ITOEC), Chongqing, China, 2022, pp. 1696-1700, doi: 10.1109/ITOEC53115.2022.9734687.
26. S. Choudhary, K. Suresh and M. S. H, "Enhanced Task Scheduling Mechanism in Logistics Based on Improvised Round Robin Technique," 2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon), Mysuru, India, 2022, pp. 1-7, doi: 10.1109/MysuruCon55714.2022.9972728.
27. M. S. Sanaj and P. M. Joe Prathap, "An Enhanced Round Robin (ERR) algorithm for Effective and Efficient Task Scheduling in cloud environment," 2020 Advanced Computing and Communication Technologies for High Performance Applications (ACCTHPA), Cochin, India, 2020, pp. 107-110, doi: 10.1109/ACCTHPA49271.2020.9213198.
28. M. Karaja, M. Ennigrou and L. B. Said, "Budget-constrained dynamic Bag-of-Tasks scheduling algorithm for heterogeneous multi-cloud environment," 2020 International Multi-Conference on: “Organization of Knowledge and Advanced Technologies” (OCTA), Tunis, Tunisia, 2020, pp. 1-6, doi: 10.1109/OCTA49274.2020.9151737.
29. A.K. M. Mashuqur Rahman Mazumder, K. M. Aslam Uddin, N. Arbe, L. Jahan and M. Whaiduzzaman, "Dynamic task scheduling algorithms in cloud computing," 2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 2019, pp. 1280-1286, doi: 10.1109/ICECA.2019.8822020.

30. Y. Yu and Y. Su, "Cloud Task Scheduling Algorithm Based on Three Queues and Dynamic Priority," 2019 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS), Shenyang, China, 2019, pp. 278-282, doi: 10.1109/ICPICS47731.2019.8942588.
31. Y. T. H. Hlaing and T. T. Yee, "Static Independent Task Scheduling on Virtualized Servers in Cloud Computing Environment," 2019 International Conference on Advanced Information Technologies (ICAIT), Yangon, Myanmar, 2019, pp. 55-59, doi: 10.1109/AITC.2019.8920865.
32. S. Kang, Bharadwaj Veeravalli, Khin Mi Mi Aung, "Dynamic scheduling strategy with efficient node availability prediction for handling divisible loads in multi-cloud systems," Journal of Parallel and Distributed Computing, Volume 113, 2018, Pages 1-16, ISSN 0743-7315, https://doi.org/10.1016/j.jpdc.2017.10.006.
33. J. Wang, S. Li, X. Zhang, K. Zhu, C. Xie and F. Wu, "Deep Reinforcement Learning Task Scheduling Method for Real-Time Performance Awareness," in IEEE Access, vol. 13, pp. 31385-31400, 2025, doi: 10.1109/ACCESS.2025.3534980.
34. J. Anand, B. Karthikeyan, "EADRL: Efficiency-aware adaptive deep reinforcement learning for dynamic task scheduling in edge-cloud environments," Results in Engineering, Volume 27, 2025, 105890, ISSN 2590-1230, https://doi.org/10.1016/j.rineng.2025.105890.
35. H. Hou, Azlan Ismail, "EETS: An energy-efficient task scheduler in cloud computing based on improved DQN algorithm," Journal of King Saud University - Computer and Information Sciences, Volume 36, Issue 8, 2024, 102177, ISSN 1319-1578, https://doi.org/10.1016/j.jksuci.2024.102177.
36. Xinglong Pei, Penghao Sun, Yuxiang Hu, Dan Li, Le Tian, Ziyong Li, "Multi-resource interleaving for task scheduling in cloud-edge system by deep reinforcement learning," Future Generation Computer Systems, Volume 160, 2024, Pages 522-536, ISSN 0167-739X, https://doi.org/10.1016/j.future.2024.06.033.
37. Jiahui Pan, Yi Wei, "A deep reinforcement learning-based scheduling framework for real-time workflows in the cloud environment," Expert Systems with Applications, Volume 255, Part D, 2024, 124845, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2024.124845.
38. D. Changzhi, C. Rong and T. Chaomu, "Design of Cloud Task Scheduling Simulation Software Based on Multi-Agent Deep Reinforcement Learning," 2024 IEEE 7th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chongqing, China, 2024, pp. 650-655, doi: 10.1109/ITNEC60942.2024.10733268.
39. Huanhuan Hou, Siti Nuraishah Agos Jawaddi, Azlan Ismail, "Energy efficient task scheduling based on deep reinforcement learning in cloud environment: A specialized review," Future Generation Computer Systems, Volume 151, 2024, Pages 214-231, ISSN 0167-739X, https://doi.org/10.1016/j.future.2023.10.002.
40. Jayanetti, S. Halgamuge and R. Buyya, "Multi-Agent Deep Reinforcement Learning Framework for Renewable Energy-Aware Workflow Scheduling on Distributed Cloud Data Centers," in IEEE Transactions on Parallel and Distributed Systems, vol. 35, no. 4, pp. 604-615, April 2024, doi: 10.1109/TPDS.2024.3360448.
41. N. K. Pandey, A. K. Mishra, N. B. Bahadure, A. N. Shukla, M. Diwakar and A. Dumka, "DRL-iCloud: Deep Reinforcement Learning Based Real-Time Task Scheduling in Integrated Cloud," 2024 International Conference on Control, Computing, Communication and Materials (ICCCCM), Prayagraj, India, 2024, pp. 247-250, doi: 10.1109/ICCCCM61016.2024.11039986.
42. X. Wang, Lin Zhang, Yongkui Liu, Chun Zhao, "Logistics-involved task scheduling in cloud manufacturing with offline deep reinforcement learning," Journal of Industrial Information Integration, Volume 34, 2023, 100471, ISSN 2452-414X, https://doi.org/10.1016/j.jii.2023.100471.
43. H. Mahmoud, M. Thabet, M. H. Khafagy and F. A. Omara, "Multiobjective Task Scheduling in Cloud Environment Using Decision Tree Algorithm," in IEEE Access, vol. 10, pp. 36140-36151, 2022, doi: 10.1109/ACCESS.2022.3163273.
44. X. Wang, Lin Zhang, Yongkui Liu, Feng Li, Zhen Chen, Chun Zhao, Tian Bai, "Dynamic scheduling of tasks in cloud manufacturing with multi-agent reinforcement learning," Journal of Manufacturing Systems, Volume 65, 2022, Pages 130-145, ISSN 0278-6125, https://doi.org/10.1016/j.jmsy.2022.08.004.
45. L. Ran, X. Shi and M. Shang, "SLAs-Aware Online Task Scheduling Based on Deep Reinforcement Learning Method in Cloud Environment," 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Zhangjiajie, China, 2019, pp. 1518-1525, doi: 10.1109/HPCC/SmartCity/DSS.2019.00209.
46. S. Chawla and A. Kaur, "Fault-Tolerant Heuristic Task Scheduling Algorithm for Efficient Resource Utilization in Cloud Computing," 2024 International Conference on Automation and Computation (AUTOCOM), Dehradun, India, 2024, pp. 131-135, doi: 10.1109/AUTOCOM60220.2024.10486076.
47. W. Chongdarakul and N. Aunsri, "Heuristic Scheduling Algorithm for Workflow Applications in Cloud-Fog Computing Based on Realistic Client Port Communication," in IEEE Access, vol. 12, pp. 134453-134485, 2024, doi: 10.1109/ACCESS.2024.3462518.
48. Boroumand, Ali, Hosseini Shirvani, Mirsaeid, Motameni, Homayun, 2024/11/27, "A heuristic task scheduling algorithm in cloud computing environment: an overall cost minimization approach", Vol. 28, doi: 10.1007/s10586-024-04843-3.
49. S. Lipsa, R. K. Dash, N. Ivković and K. Cengiz, "Task Scheduling in Cloud Computing: A Priority-Based Heuristic Approach," in IEEE Access, vol. 11, pp. 27111-27126, 2023, doi: 10.1109/ACCESS.2023.3255781.

50. R. NoorianTalouki, Mirsaeid Hosseini Shirvani, Homayun Motameni, “A heuristic-based task scheduling algorithm for scientific workflows in heterogeneous cloud computing platforms,” Journal of King Saud University - Computer and Information Sciences, Volume 34, Issue 8, Part A, 2022, Pages 4902-4913, ISSN 1319-1578, https://doi.org/10.1016/j.jksuci.2021.05.01
51. K. M. S. U. Bandaranayake, K. P. N. Jayasena and B. T. G. S. Kumara, "An Efficient Task Scheduling Algorithm using Total Resource Execution Time Aware Algorithm in Cloud Computing," 2020 IEEE International Conference on Smart Cloud (SmartCloud), Washington, DC, USA, 2020, pp. 29-34, doi: 10.1109/SmartCloud49737.2020.00015.
52. K. P. N. Jayasena, K. M. S. U. Bandaranayake and B. T. G. S. Kumara, "TRETA - A Novel Heuristic Based Efficient Task Scheduling Algorithm in Cloud Environment," 2020 IEEE REGION 10 CONFERENCE (TENCON), Osaka, Japan, 2020, pp. 812-817, doi: 10.1109/TENCON50793.2020.9293787.
53. Weiwei Lin, Weiqi Wang, Wentai Wu, Xiongwen Pang, Bo Liu, Ying Zhang, “A heuristic task scheduling algorithm based on server power efficiency model in cloud environments,” Sustainable Computing: Informatics and Systems, Volume 20, 2018, Pages 56-65, ISSN 2210-5379, https://doi.org/10.1016/j.suscom.2017.10.007
54. C. Sudhakar, M. Agroya and T. Ramesh, "Enhanced Hyper-Heuristic Scheduling Algorithm for Cloud," 2018 International Conference on Computing, Power and Communication Technologies (GUCON), Greater Noida, India, 2018, pp. 611-616, doi: 10.1109/GUCON.2018.8674941.
55. S. Devipriya and C. Ramesh, "Improved Max-min heuristic model for task scheduling in cloud," 2018 International Conference on Green Computing, Communication and Conservation of Energy (ICGCE), Chennai, India, 2018, pp. 883-888, doi: 10.1109/ICGCE.2013.6823559.
56. B. M. Hasani Zade, Najme Mansouri, Mohammad Masoud Javidi, “An improved beluga whale optimization using ring topology for solving multi-objective task scheduling in cloud,” Computers & Industrial Engineering, Volume 200, 2025, 110836, ISSN 0360-8352, https://doi.org/10.1016/j.cie.2024.110836
57. Jeng-Shyang Pan, Na Yu, Shu-Chuan Chu, An-Ning Zhang, Bin Yan, Junzo Watada, “Innovative Approaches to Task Scheduling in Cloud Computing Environments Using an Advanced Willow Catkin Optimization Algorithm,” Computers, Materials and Continua, Volume 82, Issue 2, 2025, Pages 2495-2520, ISSN 1546-2218, https://doi.org/10.32604/cmc.2024.058450
58. Yan Han, “Cloud computing task scheduling based on particle swarm optimization algorithm,” Procedia Computer Science, Volume 261, 2025, Pages 1349-1355, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2025.05.012
59. Cebrail Barut, Gungor Yildirim, Yetkin Tatar, “An intelligent and interpretable rule-based metaheuristic approach to task scheduling in cloud systems,” Knowledge-Based Systems, Volume 284, 2024, 111241, ISSN 0950-7051, https://doi.org/10.1016/j.knosys.2023.111241
60. P. Tamilarasu, G. Singaravel, “Quality of service aware improved coati optimization algorithm for efficient task scheduling in cloud computing environment,” Journal of Engineering Research, Volume 12, Issue 4, 2024, Pages 768-780, ISSN 2307-1877, https://doi.org/10.1016/j.jer.2023.09.024.
61. Laith Abualigah, Ahmad MohdAziz Hussein, Mohammad H. Almomani, Raed Abu Zitar, Hazem Migdady, Ahmed Ibrahim Alzahrani, Ayed Alwadain, “Improved synergistic swarm optimization algorithm to optimize task scheduling problems in cloud computing,” Sustainable Computing: Informatics and Systems, Volume 43, 2024, 101012, ISSN 2210-5379, https://doi.org/10.1016/j.suscom.2024.101012.
62. P. Pabitha, K. Nivitha, C. Gunavathi, B. Panjavarnam, “A chameleon and remora search optimization algorithm for handling task scheduling uncertainty problem in cloud computing,” Sustainable Computing: Informatics and Systems, Volume 41, 2024, 100944, ISSN 2210-5379, https://doi.org/10.1016/j.suscom.2023.100944.
63. Chirag Chandrashekar, Pradeep Krishnadoss, Vijayakumar Kedalu Poornachary, Balasundaram Ananthakrishnan, “MCWOA Scheduler: Modified Chimp-Whale Optimization Algorithm for Task Scheduling in Cloud Computing,” Computers, Materials and Continua, Volume 78, Issue 2, 2024, Pages 2593-2616, ISSN 1546-2218, https://doi.org/10.32604/cmc.2024.046304.
64. Z. A. Khan, I. A. Aziz, N. A. B. Osman and S. Nabi, "Parallel Enhanced Whale Optimization Algorithm for Independent Tasks Scheduling on Cloud Computing," in IEEE Access, vol. 12, pp. 23529-23548, 2024, doi: 10.1109/ACCESS.2024.3364700.
65. Z. Cui, T. Zhao, L. Wu, A. K. Qin and J. Li, "Multi-Objective Cloud Task Scheduling Optimization Based on Evolutionary Multi-Factor Algorithm," in IEEE Transactions on Cloud Computing, vol. 11, no. 4, pp. 3685-3699, Oct.-Dec. 2023, doi: 10.1109/TCC.2023.3315014.
66. Sudheer Mangalampalli, Ganesh Reddy Karri, Utku Kose, “Multi objective trust aware task scheduling algorithm in cloud computing using whale optimization,” Journal of King Saud University - Computer and Information Sciences, Volume 35, Issue 2, 2023, Pages 791-809, ISSN 1319-1578, https://doi.org/10.1016/j.jksuci.2023.01.016.
67. H. Zhang and R. Jia, "Application of Chaotic Cat Swarm Optimization in Cloud Computing Multi Objective Task Scheduling," in IEEE Access, vol. 11, pp. 95443-95454, 2023, doi: 10.1109/ACCESS.2023.3311028.
68. Raja Masadeh, Nesreen Alsharman, Ahmad Sharieh, Basel A. Mahafzah, Arafat Abdulrahman, “Task scheduling on cloud computing based on sea lion optimization algorithm,” International Journal of Web Information Systems, Volume 17, Issue 2, 2021, Pages 99-116, ISSN 1744-0084, https://doi.org/10.1108/IJWIS-11-2020-0071.
69. X. Chen et al., "A WOA-Based Optimization Approach for Task Scheduling in Cloud Computing Systems," in IEEE Systems Journal, vol. 14, no. 3, pp. 3117-3128, Sept. 2020, doi: 10.1109/JSYST.2019.2960088.

70. Mohaymen Selselejoo, HamidReza Ahmadifar, “DT-GWO: A hybrid decision tree and GWO-based algorithm for multi-objective task scheduling optimization in cloud computing,” Sustainable Computing: Informatics and Systems, Volume 47, 2025, 101138, ISSN 2210-5379, https://doi.org/10.1016/j.suscom.2025.101138.
71. N. Omran Alkaam, A. Bakar Md Sultan, M. B. Hussin and K. Yatim Sharif, "Hybrid Henry Gas-Harris Hawks Comprehensive-Opposition Algorithm for Task Scheduling in Cloud Computing," in IEEE Access, vol. 13, pp. 12956-12965, 2025, doi: 10.1109/ACCESS.2025.3530860.
72. Ipsita Behera, Srichandan Sobhanayak, “HTSA: A novel hybrid task scheduling algorithm for heterogeneous cloud computing environment,” Simulation Modelling Practice and Theory, Volume 137, 2024, 103014, ISSN 1569-190X, https://doi.org/10.1016/j.simpat.2024.103014.
73. Shasha Zhao, Huanwen Yan, Qifeng Lin, Xiangnan Feng, He Chen, Dengyin Zhang, “Hybrid Hierarchical Particle Swarm Optimization with Evolutionary Artificial Bee Colony Algorithm for Task Scheduling in Cloud Computing,” Computers, Materials and Continua, Volume 78, Issue 1, 2024, Pages 1135-1156, ISSN 1546-2218, https://doi.org/10.32604/cmc.2024.045660.
74. B. Saemi, A. A. R. Hosseinabadi, A. Khodadadi, S. Mirkamali and A. Abraham, "Solving Task Scheduling Problem in Mobile Cloud Computing Using the Hybrid Multi-Objective Harris Hawks Optimization Algorithm," in IEEE Access, vol. 11, pp. 125033-125054, 2023, doi: 10.1109/ACCESS.2023.3329069.
75. 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," in IEEE Access, vol. 10, pp. 17803-17818, 2022, doi: 10.1109/ACCESS.2022.3149955.
76. K. Li, L. Jia and X. Shi, "Research on Cloud Computing Task Scheduling Based on PSOMC," in Journal of Web Engineering, vol. 21, no. 6, pp. 1749-1766, September 2022, doi: 10.13052/jwe1540-9589.2161.
77. N. Manikandan, P. Divya, S. Janani, “BWFSO: Hybrid Black-widow and Fish swarm optimization Algorithm for resource allocation and task scheduling in cloud computing,” Materials Today: Proceedings, Volume 62, Part 7, 2022, Pages 4903-4908, ISSN 2214-7853, https://doi.org/10.1016/j.matpr.2022.03.535.
78. S. Velliangiri, P. Karthikeyan, V.M. Arul Xavier, D. Baswaraj, “Hybrid electro search with genetic algorithm for task scheduling in cloud computing,” Ain Shams Engineering Journal, Volume 12, Issue 1, 2021, Pages 631-639, ISSN 2090-4479, https://doi.org/10.1016/j.asej.2020.07.003.
79. M.S. Sanaj, P.M. Joe Prathap, “An efficient approach to the map-reduce framework and genetic algorithm-based whale optimization algorithm for task scheduling in cloud computing environment,” Materials Today: Proceedings, Volume 37, Part 2, 2021, Pages 3199-3208, ISSN 2214-7853, https://doi.org/10.1016/j.matpr.2020.09.064
80. M. Abd Elaziz, Shengwu Xiong, K.P.N. Jayasena, Lin Li, “Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution,” Knowledge-Based Systems, Volume 169, 2019, Pages 39-52, ISSN 0950-7051, https://doi.org/10.1016/j.knosys.2019.01.023.
81. S. Pang, W. Li, H. He, Z. Shan and X. Wang, "An EDA-GA Hybrid Algorithm for Multi-Objective Task Scheduling in Cloud Computing," in IEEE Access, vol. 7, pp. 146379-146389, 2019, doi: 10.1109/ACCESS.2019.2946216.