International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 17 (2025) pp. 466–485

Article

Enhancing Cloud Computing Performance for Fast Online Services Delivery Using a Novel Hybrid Task Scheduling Algorithm

Kibreab Adane [1,*], Kedamo Mohammed [2] and Keyre Kemal [2]

[1] Faculty of Computing & Software Engineering, Arba Minch University, Arba Minch 21, Ethiopia
[2] Department of Information Technology, Werabe University, Werabe 46, Ethiopia
* Corresponding author: kibreab.adane@amu.edu.et; [0000-0002-3021-5059]

Received date: 20 July 2024; Accepted date: 20 March 2025; Published online: 29 May 2025

Abstract: Forwarding many tasks to the cloud server simultaneously degrades cloud computing performance in this regard, time-dependent sectors like banks, healthcare, airlines, online shopping, and more suffer from slow delivery of cloud computing services unless a suitable task scheduling technique is implemented. The existing literature-based evidence reveals that effective task scheduling in cloud computing optimizes resource usage, boosts fault tolerance, facilitates scalability, lowers costs, and reduces energy consumption. However, identifying task-scheduling algorithms that fulfill the stated benefits is a hot research topic. To address these concerns, the study explored the type of task scheduling algorithms utilized, the dataset source, the number of tasks given, the number of cloud hosts, the simulation toolkits, the kind of evaluation metrics used, the number of virtual machines, and the scheduling techniques that performed best overall from 36 recent, relevant and reliable research works; applied one of the top performing task scheduling algorithms in most reviewed studies i.e., Genetic Algorithm(GA) along with task scheduling algorithm used in none of the reviewed research works i.e., Earliest Deadline First (EDF) as novel contributions, Compares the results of the proposed approach with another study, which employed the same datasets and evaluation metrics. The overall experiment results show that the proposed approach outperforms GA and GWO algorithms in terms of minimizing Makespan across all workloads and optimizing resource utilization on all workloads, except on workloads 300 and 500, where it performs less than GWO algorithms. GWO algorithm is slightly performing better than the proposed approach on workloads 500, 600, 700, 800, 900, and 1000 in maximizing throughput while performing less than the proposed approach on workloads 100, 200, and 300 in maximizing throughput. Compared to the proposed approach (GA-EDF combination) and GWO algorithm, the GA task scheduling algorithms demonstrate poor performance based on Makespan, average resource utilization, and throughput.

Keywords: cloud computing; task scheduling; makespan; CloudSim; throughput; resource utilization

1. Introduction

Cloud computing is a cutting-edge technology that permits online users to obtain reliable Internet services at any time, from any geographical location, via diverse platforms or devices at low cost [1,2]. However, the availability of diverse computing nodes with resources of varied types in cloud data centers has negative impacts on fast service delivery to online users [3] unless the intervention of a suitable task scheduling algorithm is made to address these concerns.

Cloud computing enables quick delivery of online services and follows pay-per-use principle [4]. Moreover, the proper use of resources, reduction of costs and energy consumption, and enhancement of fault tolerance require efficient task scheduling [5]. Task scheduling assigns the task to multiple cloud computing resources to prevent overload or under-load virtual machines [6,7].

Task/ job scheduling algorithms are classified into Static and dynamic. In static scheduling, tasks are assigned to existing resources manually and all task detail is known before the schedule [8]. It does not check the current state of the resource at the time of assigning tasks, which can lead to overloading the system and degrading its performance. Dynamic task scheduling is preferred by most researchers [9] because it addresses the aforementioned concerns. Inefficient task scheduling could result in overloading of some Virtual Machines (VMs) while others are idle or under-loaded [10,11].

The core challenges associated with the task scheduling algorithms include: i) Dynamic adaptability to changes; ii) Not complying with QoS demands or deadlines, iii) lacking fault-tolerance; iv) algorithm complexity for proper implementation and operation [12–14]. The task scheduling algorithms have their intrinsic properties in doing scheduling tasks. For instance, the First Come Frist Serve Algorithm (FCFS), assigns tasks/jobs to Virtual Machines in the order determined by their arrival time and implemented via a FIFO queue. Shortest Job First (SJF), first prioritize tasks/jobs having the smallest processing time are prior first. MAX-MIN (MM), ordering tasks based on the completion time [15]. Round Robin (RR), follows the time-slicing method for each node to assign a task. Particle Swarm Opt. (PSO), looking for a viable schedule within that framework based on a pre-defined fitness parameter. In Ant Colony Opt. (ACO), the behavior of real ant colonies is imitated and can be used in complex or dynamic environments [16].

Indeed, effective task scheduling in cloud computing optimizes resource usage, boosts fault tolerance, facilitates scalability, lowers costs, and encourages energy efficiency. Due to these facts, the identification of task-scheduling algorithms that fulfill most of the aforementioned benefits remains a hot research topic.

The core rationale of the study is that in the reviewed studies, several task scheduling algorithms are used in cloud computing environment, despite not all of them performing at the top level in this regard. The Scheduling Algorithm, like the Genetic Algorithm (GA), is a frequently used and top-performing task scheduling algorithm in the different reviewed studies (see literature review section). The recent study [1] reveals that the Grey Wolf Optimization Algorithm (GWO) and Genetic Algorithms (GA) offer distinct advantages; they can be used to improve the accuracy and efficiency of the optimization process in a variety of optimization issues depending on specific requirements, they show faster convergence in big scheduling problems and they are efficient in reducing energy usage, computational cost, and makespan, as well as effective in avoiding local optima [1]. As stated in [2] to obtain the near-optimal solutions, meta-heuristic algorithms employ a random search method. Numerous common meta-heuristics are used for task scheduling, including simulated annealing (SA), particle swarm (PSO), genetic algorithms (GA), ant colony optimization (ACO), and more. However, these algorithms have multiple search methods, randomness issues, limited global search capabilities, and less convergence in the late iteration, which prevents them from finding the local optimum search solution. Additionally, there is an imbalance between local and global search. The majority of issues with meta-heuristic algorithms are resolved by the grey wolf optimizer (GWO). Compared to the PSO technique, the GWO mechanism uses less memory because it only needs the position of one vector. Furthermore, unlike the PSO algorithm, which selects one best solution by all particles, GWO depends on the three best solutions to prevent it from falling into the local optimal solution. Furthermore, compared to GA algorithms, the GWO algorithm has fewer parameters, meaning it is simpler and uses less energy and processing time. Compared to Ant Colony optimization, which ignores the dynamic aspect of computing resources, the GWO algorithm is more dynamic. Compared to other meta-heuristic algorithms, GWO has recently become popular because of its many advantages, including ease in implementation, fewer parameters that result in minimal complexity time and energy usage, and convergence during execution [2]. These are one of the core rationale for including the GWO and GA scheduling algorithms in our study.

According to recent study [3] the Earliest Deadline First (EDF) is the best uniprocessor scheduling algorithm. This indicates that no other scheduling method can realistically schedule a collection of tasks if they cannot be scheduled using EDF. The EDF algorithm selects the job or jobs with the closest deadlines for execution at each moment in time. The following guidelines govern how EDF is implemented on uniform parallel machines: Higher priority jobs are executed on faster processors, and while there are fewer than m jobs active, they must execute on the fastest processor while the slowest are idled. No processor is idle while there are active jobs waiting to execute [3]. EDF is a suitable algorithm for online or real time scheduling on uniform multiprocessors. However, because of the enormous swings brought on by jobs with comparatively closer deadlines being completed or arriving, their implementation suffers from a large number of migrations. The cost of task migration could be very expensive. For instance, a migration carried out in a loosely linked system, such a cluster of workstations, is done so slowly that the overload brought on by an excessive migration may not be acceptable. Another drawback of EDF is that when it is overwhelmed, its behavior becomes erratic. As a result, when EDF is overloaded, its performance declines to the point where it is not suitable for usage [3]. The

aforementioned limitations of EDF could be addressed when combined with the genetic algorithm (GA) because GA shows faster convergence in big scheduling problems and is efficient in reducing energy usage, computational cost, and makespan, as well as effective in avoiding local optima [1].

The review findings reveal that each task scheduling algorithm has its limitations. Taking into account their intrinsic properties in optimizing the task-scheduling process specifically in reducing high computational complexity, slow convergence, and inefficiency in dynamic environments, the study selected the genetic algorithm (GA), grey wolf optimizer (GWO), and the Earliest Deadline First (EDF) task scheduling algorithms. Moreover, given that GA is a popular and highly effective task scheduling algorithm in a number of reviewed studies, that EDF is suitable for real-time (online) scheduling, that higher-priority jobs are executed on faster processors in EDF [3], and that none of the 36 reviewed studies used the EDF task scheduling algorithm, the study proposed hybrid task scheduling algorithms that combine GA and EDF to fill the literature gaps. In short, the major contributions of the study are presented as follows.

- Used structured guidelines to excavate open issues and best current practices regarding different task scheduling algorithms in cloud computing field from recent and pertinent research articles extracted from SCOPUS and WoS Databases, published between 2020 and 2024.
- Explored the type of task scheduling algorithms utilized, the dataset source, the number of tasks/workload, the number of cloud hosts, the simulation toolkits, the kind of evaluation metrics used, the number of virtual machines, and the scheduling techniques that performed best overall.
- Employed the frequently top-performing task scheduling algorithm in the different reviewed studies like Genetic Algorithm (GA) with task scheduling algorithms that is not incorporated in the reviewed studies the Earliest Deadline First (EDF) as novel practical contributions. This is done due to better-performing task scheduling algorithms most reviewed studies are those applied in a hybrid approach. The viability of the EDF algorithm in online or real time task scheduling was indicated in [3]. None of the rigorously reviewed studies applied GA and EDF as a hybrid approach considering the individual capabilities of these task scheduling algorithms.
- Compared the proposed approach performances against the counterpart using popular evaluation metrics like resource utilization, Makespan, and throughput on the same datasets.

2. Literature Review

The study rigorously assessed recent and pertinent related research works focused on task/job scheduling in CC (Cloud Computing). To achieve this aim, keyword-based search strategy is formatted like (“Task Scheduling Algorithm” AND “Cloud Computing”) OR (“Cloud based” AND “Task Scheduling”) OR “Job Scheduling Algorithm” AND “Cloud Computing”) OR (“Task Scheduling” AND “Cloud Environment”, (“Load Balancing” AND “Cloud Computing”). These keywords were sent via Google engine and Mendeley literature search at the first stage and only research publications from 2020 to 2024 were downloaded to look for the current gaps and best practices regarding task scheduling and cloud computing. Accordingly, the study was able to retrieve 53 different research works after checking Journal indexing and abstracting in SCOPUS OR Web of Science OR Both considering global acceptance of these databases in terms of peer-review, reliability, and relevance. After checking the suitability of the abstract and full contents as well as removing duplicates 36 research works were screened out, assessed using structured guidelines, and presented in Table 1.

To enhance Cloud computing performance, Ref. [17] implemented ANN- PSO, PSO, IBPSO-LBS, and Heuristic-FSA task scheduling algorithms, used CloudSim toolkit, 15 number of hosts, 1000 to 5000 tasks, 15 VMs, used evaluation metrics (Resource Utilization, Response Time, Makespan), and the authors' findings reveal that ANN-BPSO scored the top performance despite not revealing the name of the dataset used. Ref. [18] implemented Q-Learning task scheduling algorithms, used CloudSim toolkit, 20 number of hosts, 1000 tasks, 50 VMs, used evaluation metrics (Response Time Energy Consumption), and the authors' findings reveal that Q-Learning scored the top performance despite not revealing the name of the dataset used. Ref. [19] implemented LCFP, SJF, CCSA, GA, HSLJF task scheduling algorithms, used CloudSim toolkit, 2 number of hosts, 500 to 2000 tasks, VMs (16, 32, and 64), used evaluation metrics (Makespan, Response Time, Throughput, and Flow Time), and the authors' findings reveal that GA scored the top performance despite not revealing the name of the dataset used. Ref. [20] implemented DRALB task scheduling algorithms, used CloudSim toolkit, number of hosts (0 to 200), tasks (40, 80, 120, 160, 200), 10 VMs, used evaluation metrics (Rate of SLA Violation, Response Time, Load Balancing, Resource Utilization), and the authors' findings reveal that DRALB scored the top performance despite not revealing the name of the dataset used, see Table 1.

Ref. [21] implemented Min-Min and Max-Min task scheduling algorithms, used GoCJ dataset, CloudSim toolkit, tasks (100 to 1000), VMs (1 to 12), used evaluation metrics (Resource Utilization,

Makespan, AVG. Waiting Time, Task Failure Rate), and the authors' findings reveal that Min-Min scored the top performance despite not revealing the number of hosts used. Ref. [22] implemented GWO, PSO, and GA task scheduling algorithms, used GoCJ & Synthetic dataset, tasks (100 to 1200), 10, VMs, used evaluation metrics (Makespan, Throughput, Convergence Time) despite not revealing the number of hosts and Simulation toolkit used. Ref. [23] implemented PSO, MVO, and EMVO task scheduling algorithms, used GoCJ dataset, MATLABv.8.5.0 toolkit, tasks (100 to 1000), VMs (10 to 60), used evaluation metrics (Throughput, Makespan, Resource Utilization), and the authors' findings reveal that EMVO Scored the top performance, despite not revealing the number of hosts. Ref. [24] implemented ANN–FOA, Tabu search, PSO, GA, & simulated annealing task scheduling algorithms, used Fog-cloud smart grid/CloudSim toolkit, 2 hosts per fog node, tasks (50, 100, 150), 6 VMs, used evaluation metrics (Allocated memory, execution time, latency, and cost), and the authors' findings reveal that ANN–FOA Scored the top performance, despite not revealing the name of the dataset used, see Table 1.

Table 1. Guideline for Research Gaps Identification.

Evaluation Criteria	**Ref. [17], 2022**	**Ref. [18], 2020**	**Ref. [19], 2021**	**Ref. [20], 2021**
Task scheduling algorithm used	ANN-PSO, PSO, IBPSO-LBS, Heuristic-FSA	Q-Learning	LCFP, SJF, CCSA, GA,HSLJF	DRALB
Name of Dataset	Not Found	Not Found	Not Found	Not Found
Simulation tool	CloudSim	CloudSim	CloudSim	CloudSim
No. of Cloud hosts	15	20	2	0 to 200
No of tasks	1000 to 5000	1000	500 to 2000	40, 80, 120, 160, 200
Evaluation Metrics used	Resource Utilization, Response Time, Makespan,	Response Time Energy Consumption ,	Makespan, Response Time, Throughput, & Flow Time	Rate of SLA Violation, Response Time, Load Balancing, Resource Utilization.
No. Virtual Machines(VM)	15	50	16, 32, & 64	10
Better performing scheduling algorithm(s)	ANN-BPSO	Q-Learning	GA	DRALB
Evaluation Criteria	**Ref. [21], 2022**	**Ref. [22], 2023**	**Ref. [23], 2021**	**Ref. [24], 2022**
Task scheduling algorithm used	Min-Min, Max-Min	GWO, PSO, GA	PSO, MVO, EMVO,	ANN–FOA, Tabu search, PSO, GA, & simulated annealing
Name of Dataset	GoCJ dataset	GoCJ & Synthetic dataset	GoCJ dataset	Not Found
Simulation tool	CloudSim	Not Found	MATLABv.8.5.0	Fog-cloud smart grid/CloudSim
No. of Cloud hosts	Not Found	Not Found	Not Found	2 hosts per fog node
No of tasks	100 to 1000	100 to 1200	100 to 1000	50, 100, 150
Evaluation Metrics used	Resource Utilization, Makespan, AVG. Waiting Time, Task Failure Rate	Makespan, Throughput, Convergence Time	Throughput, Makespan, Resource Utilization	Allocated memory, execution time, latency, and cost
No. Virtual Machines(VM)	1 to 12	Not Found	10 to 60	6
Better performing scheduling algorithm(s)	Min-Min	Not Found	EMVO	ANN–FOA

Ref. [25] implemented DRRHA, DRRHA-SJF, and DRRHA-FCFS task scheduling algorithms, used NASA dataset, CloudSim Plus toolkit, tasks (10 to 400), 1VMs, used evaluation metrics (Response Time, Context switches, Wait Time, Turnaround time), and the authors' findings reveal that DRRHA Scored the

top performance, despite not revealing the number of hosts. Ref. [26] implemented BAAEQRL, DBA, BA, PSO, CS, GA, HS, and DE task scheduling algorithms, used Synthetic Workloads and Standard Parallel Work Loads dataset, CloudSim toolkit, tasks (1000 to 5000), 20 VMs, used evaluation metrics (Cost, Makespan), and the authors' findings reveal that BAAEQRL Scored the top performance, despite not revealing the number of hosts. Ref. [27] implemented DRALBA, Dynamic Max-Min, DLBA, RALBA, and PSSELB task scheduling algorithms, used GoCJ, HCSP, and Synthetic workload dataset, CloudSim toolkit, tasks (500-2500), VMs (16 to 64, 128 to 256, 512 to 1024), used evaluation metrics (Throughput, AVG Resource Utilization, Makespan, & AVG Response Time), and the authors' findings reveal that DRALBA Scored the top performance, despite not revealing the number of hosts. Ref. [28] implemented PLB, Max-Min, FCFS, Min-Min, RR, and ACO task scheduling algorithms, used TPC-VMS &TPC-DI dataset, CloudSim toolkit, tasks (100 to 1000), VMs (5 to 10), used evaluation metrics (AVG Response Time, Makespan, bandwidth, & Resource Utilization), and the authors' findings reveal that PLB Scored the top performance, see Table 2.

Ref. [29] implemented H3CSA, HEFT-T, HEFT-L, HEFT-B, GA, and PSO task scheduling algorithms, used CloudSim toolkit, tasks (5 to 100), VMs (3 to 8), used evaluation metrics (Makespan), and the authors' findings reveal that H3CSA Scored the top performance, despite not revealing the number of hosts and name of datasets used. Ref. [30] implemented Genetic Algorithm (GA), IWD, WOA, MFO, and PSO task scheduling algorithms, used MATLAB toolkit, tasks (20, 40, 60), used evaluation metrics (Best, Medium, Worst speeds), and the authors' findings reveal that GA Scored the top performance, despite not revealing the number of hosts, number of VMs, and name of datasets used. Ref. [31] implemented RR, GA, and Tournament Sel. GA (TS-GA) task scheduling algorithms, used CloudSim toolkit, tasks (500-2500), 8 number of hosts, 8 VMs, used evaluation metrics (Execution Cost, Completion Time, Resource Utilization), and the authors' findings reveal that TS-GA Scored the top performance, despite not revealing the name of datasets used. Ref. [32] implemented DAIRS with Q-Learning, IDE task scheduling algorithms, used NASA dataset, tasks (1000 to 100000), VMs (10, 20. 50), used evaluation metrics (Resource Utilization, Waiting Time, Delay Execution), and the authors' findings reveal that DAIRS with Q-Learning Scored the top performance, despite not revealing the number of hosts and simulation toolkit used, see Table 2.

Table 2. Guideline for Research Gaps Identification.

Evaluation Criteria	**Ref. [25], 2021**	**Ref. [26], 2022**	**Ref. [27], 2021**	**Ref. [28], 2021**
Task scheduling algorithm used	DRRHA, DRRHA-SJF, DRRHA-FCFS,	BAAEQRL, DBA, BA, PSO, CS, GA, HS, DE	DRALBA, Dynamic Max-Min, DLBA, RALBA, & PSSELB	PLB, Max-Min, FCFS, Min-Min, RR, ACO
Name of Dataset	NASA dataset	Synthetic Workloads & Standard Parallel Work Loads	GoCJ, HCSP, and Synthetic workload	TPC-VMS &TPC-DI
Simulation tool	CloudSim Plus	CloudSim	CloudSim	CloudSim
No. of Cloud hosts	Not Found	Not Found	Not Found	1, 200, 400,& 1000
No of tasks	10 to 400	1000 to 5000	500-2500	100 to 1000
Evaluation Metrics used	Response Time, Context switches, Wait Time, Turnaround time	Cost, Makespan	Throughput, AVG Resource Utilization, Makespan, & AVG Response Time	AVG Response Time, Makespan, bandwidth,& Resource Utilization
No. Virtual Machines(VM)	1	20	16 to 64, 128 to 256, 512 to 1024	5 to 10
Better performing scheduling algorithm(s)	DRRHA	BAAEQRL	DRALBA	PLB
Evaluation Criteria	**Ref. [29], 2021**	**Ref. [30], 2022**	**Ref. [31], 2021**	**Ref. [32], 2021**
Task scheduling algorithm used	H3CSA, HEFT-T, HEFT-L, HEFT-B, GA, PSO	Genetic Algorithm (GA), IWD, WOA, MFO,& PSO	RR, GA, Tournament Sel. GA(TS-GA),	DAIRS with Q-Learning, IDE
Name of Dataset	Not Found	Not Found	Not Found	NASA dataset

Simulation tool	CloudSim	MATLAB	CloudSim	Not Found
No. of Cloud hosts	Not Found	Not Found	8	Not Found
No of tasks	5 to 100	20, 40, 60	25, 50, 75, 100	1000 to 100000
Evaluation Metrics used	Makespan	Best, Medium & Worst speeds	Execution Cost, Completion Time, Resource Utilization	Resource Utilization, Waiting Time, Delay Execution
No. Virtual Machines(VM)	3 to 8	Not Found	8	10, 20. 50
Better performing scheduling algorithm(s)	H3CSA	GA	TS-GA	DAIRS with Q-Learning

Ref. [33] implemented MHDNNL, PSO, and RR, task scheduling algorithms, used Python & TensorFlow toolkit, number of hosts (3 to 10), tasks (100 to1000), used evaluation metrics (Latency, Energy Consumption,), and the authors' findings reveal that MHDNNL Scored the top performance, despite not revealing the name of datasets and number of VMs. Ref. [34] implemented Ant colony opt.(ACO), PSO, and Simulated Annealing task scheduling algorithms, used MATLAB toolkit, 10 number of hosts, tasks (100 to 1000), used evaluation metrics (Delay, Energy Consumption), and the authors' findings reveal that ACO Scored the top performance, despite not revealing the name of datasets used. Ref. [35] implemented Parallel Enhanced- WOA, WAO, and PSO task scheduling algorithms, used GoCJ & HCSP Datasets, CloudSim toolkit, tasks (100 to 1000), 8 VMs, used evaluation metrics (Throughput, Makespan, Resource Utilization, Execution Time, Response Time), and the authors' findings reveal that PEWOA Scored the top performance, despite not revealing the number of hosts used. Ref. [36] implemented Seagull opt. algorithm (SOA) & Whale Opt. Algorithm (WOA), SMA, BSO task scheduling algorithms, 25 number of hosts, used CloudSim toolkit, tasks (100 to 1000), VMs (10, 150, 200, 450, 800), used evaluation metrics (Execution Time, Resource utilization, Response Time, Processing Time, Throughput), and the authors' findings reveal that WOA-SOA Scored the top performance, despite not revealing the name of datasets used, see Table 3.

Ref. [37] implemented Modified Backfilling (MBF), and Ant Colony Opt. (ACO) task scheduling algorithms, used PlanetLab dataset & GoCJ datasets, CloudSim toolkit, 48 number of hosts, tasks (200 to 1000), 60 VMs, used evaluation metrics (Makespan, Completion Time, Degree of imbalance, Energy Consumption, SLA), and the authors' findings reveal that ACO Scored the top performance. Ref. [38] implemented Art. Bee Colony(ABC), Q-learning, Max-Min, PSO, FCFS, LJF task scheduling algorithms, used GoCJ & Synthetic workload datasets, CloudSim toolkit, 20 number of hosts, tasks (1024), used evaluation metrics (Makespan, Cost, Degree of Imbalance, Throughput, AVG Resource Utilization), VMs (5 to 100) and the authors' findings reveal that ABC-Q-Learning Scored the top performance. Ref. [39] implemented ACO, GA, and GA-ACO task scheduling algorithms, used HSCP/u.c-lohi dataset, CloudSim toolkit, tasks (1024), 32 VMs, used evaluation metrics (Makespan, AVG Resource Utilization, & SLA), and the authors' findings reveal that GA-ACO Scored the top performance, despite not revealing the number of hosts used. Ref. [40] implemented ACO, PSO, and Grey Wolf Opt (GWO) task scheduling algorithms, 20 number of hosts, used CloudSim toolkit, tasks (100 to 1000), 100 VMs, used evaluation metrics (Makespan AVG Waiting Time, Resource Utilization, AVG Exec. Time, and Energy Consumption), and the authors' findings reveal that GWO Scored the top performance, despite not revealing the name of datasets and simulation toolkit used, see Table 3.

Table 3. Guideline for Research Gaps Identification.

Evaluation Criteria	**Ref. [33], 2022**	**Ref. [34], 2023**	**Ref. [35], 2024**	**Ref. [36], 2023**
Task scheduling algorithm used	MHDNNL, PSO, RR,	Ant colony opt.(ACO), PSO, Simulated Annealing	Parallel Enhanced-WOA, WAO, PSO	Seagull opt. algorithm (SOA) & Whale Opt. Algorithm(WOA), SMA, BSO
Name of used dataset	Not Found	Not Found	GoCJ & HCSP datasets	Not Found
Simulation tool	Python & TensorFlow	MATLAB	CloudSim	CloudSim
No. of Cloud hosts	3 to 10	10	Not Found	25

No of tasks	100 to1000	100 to 1000	100 to 1000	100 to 1000
Evaluation Metrics used	Latency Energy Consumption,	Delay, Energy Consumption	Throughput, Makespan, Resource Utilization, Execution Time, Response Time	Execution Time, Resource utilization, Response Time, Processing Time, Throughput
No. Virtual Machines(VM)	Not Found	Not Found	32& 64 for small jobs 128&256 for medium Jobs 512& 1024 for large jobs	10, 150, 200, 450, 800
Better Performing Scheduling Algorithm	MHDNNL	ACO	PEWOA	WOA-SOA
Evaluation Criteria	**Ref. [37], 2022**	**Ref. [38], 2022**	**Ref. [39], 2021**	**Ref. [40], 2022**
Task scheduling algorithm used	Modified Backfilling (MBF), Ant Colony Opt. (ACO),	Art. Bee Colony(ABC), Q-learning, Max-Min, PSO, FCFS, LJF	ACO, GA, GA-ACO	ACO, PSO, Grey Wolf Opt (GWO)
Name of used dataset	PlanetLab dataset & GoCJ dataset	GoCJ & Synthetic workload datasets	HSCP/u.c-lohi dataset	Not Found
Simulation tool	CloudSim	CloudSim	CloudSim	Not Found
No. of Cloud hosts	48	20	Not Founds	20
No of tasks	200 to 1000	200 to 1000	1024	100–1000
Evaluation Metrics used	Makespan, Completion Time, Degree of imbalance, Energy Consumption, SLA,	Makespan, Cost, Degree of Imbalance, Throughput ,AVG Resource Utilization	Makespan, AVG Resource Utilization, & SLA	Makespan AVG Waiting Time, Resource Utilization, AVG Exec. Time, & Energy Consumption
No. Virtual Machines(VM)	60	5 to 100	32	100
Better Performing Scheduling Algorithm	ACO	ABC-Q-Learning	GA-ACO	GWO

Ref. [41] implemented TBTS, Max-Min, EXIST, FCFS, Min-Min, SLA-MCT, LB-Max-Min, and SLA-LB task scheduling algorithms, used synthetic dataset, HCSP/u.c.hilo, GCC compiler V.4.4.4 toolkit, tasks (512, 1024, 2048), VMs (16, 32, 64), used evaluation metrics (Makespan, gain cost, Resource Utilization, Penalty cost), and the authors' findings reveal that TBTS Scored the top performance, despite not revealing the number of hosts. Ref. [42] implemented BWM-TOPSIS task scheduling algorithms, used CloudSim toolkit, tasks (100 to 1000), used evaluation metrics (Cost, Throughput, waiting time, Makespan), VMs (1, 2, 4) and the authors' findings reveal that BWM-TOPSIS Scored the top performance, despite not revealing the name of datasets and number of hosts used. Ref. [43] implemented Balancer GA(BGA), DSOS, ETA-GA, MGGS, RALBA task scheduling algorithms, used Synthetic dataset, CloudSim toolkit, 30 number of hosts, tasks (100 to 1000), 50 VMs, used evaluation metrics (AVG Resource Utilization, Makespan, Throughput), and the authors' findings reveal that BGA Scored the top performance. Ref. [44] implemented Min-Min and Max-Min task scheduling algorithms, used CloudSim toolkit, tasks (100 to 1000), used evaluation metrics (Makespan, Response Time, AVG Resource Utilization,), and the authors' findings reveal that GWO Scored the top performance, despite not revealing the name of datasets, number of hosts, and VMs, see Table 4.

Ref. [45] implemented GA, GA-IRACE, and RACE task scheduling algorithms, used CloudSim toolkit, number of hosts (4, 8, 10), tasks (100 to 900), VMs (70, 80, 100), used evaluation metrics (Execution time, Cost, Response Time), and the authors' findings reveal that GA-IRACE Scored the top performance, despite not revealing the name of datasets used. Ref. [46] implemented OPFT, Min-Min, DCTS, GTS, and K-Means clustering task scheduling algorithms, used GoCJ Dataset, CloudSim toolkit, 4 number of hosts, tasks (500,1000, 1500, 2000), used evaluation metrics (Makespan, AVG Execution Time, Workload Deviation), VMs (30, 50) and the authors' findings reveal that K-Means clustering Scored the top performance. Ref. [47] implemented GA, BF, ACO, and ACO-BF task scheduling algorithms, used, CloudSim toolkit, tasks (10 to 200), used evaluation metrics (Makespan, Energy Consumption, Resource Utilization), and the authors' findings reveal that ACO-BF Scored the top performance, despite not revealing the name of datasets, number of hosts, and VMs. Ref. [48] implemented PSO, HSO, SSO, PSO-SSO task scheduling algorithms, used CloudSim toolkit, tasks (1000), 100 VMs, used evaluation metrics (Makespan, Throughput, Execution Time), and the authors' findings reveal that HSO Scored the top performance, despite not revealing the name of datasets and number of hosts, see Table 4.

Table 4. Guideline for Research Gaps Identification.

Evaluation Criteria	**Ref. [41], 2020**	**Ref. [42], 2020**	**Ref. [43], 2021**	**Ref. [44], 2020**
Task scheduling algorithm used	TBTS, Max-Min, EXIST, FCFS, Min-Min, SLA-MCT, LB-Max-Min, SLA-LB	BWM-TOPSIS	Balancer GA(BGA), DSOS, ETA-GA, MGGS, RALBA	Min-Min, Max-Min
Name of used dataset	synthetic dataset, HCSP/u.c.hilo	Not Found	Synthetic dataset	Not Found
Simulation tool	GCC compiler V.4.4.4.	CloudSim	CloudSim	CloudSim
No. of Cloud hosts	Not Found	Not Found	30	Not Found
No of tasks	512, 1024, 2048	100 to 1000	100 to 1000	100 to 1000
Evaluation Metrics used	Makespan, Gain cost, Resource Utilization, Penalty cost	Cost, Throughput, Waiting time, Makespan	AVG Resource Utilization, Makespan, Throughput	Makespan, Response Time, AVG Resource Utilization,
No. Virtual Machines(VM)	16, 32, 64	1, 2, 4	50	Not Found
Better Performing Scheduling Algorithm	TBTS	BWM-TOPSIS	BGA	Not Found
Evaluation Criteria	**Ref. [45], 2022**	**Ref. [46], 2021**	**Ref. [47], 2021**	**Ref. [48], 2023**
Task scheduling algorithm used	GA, GA-IRACE, RACE	OPFT, Min-Min, DCTS, GTS, K-Means clustering	GA, BF, ACO, ACO-BF	PSO, HSO, SSO, PSO-SSO
Name of used dataset	Not Found	GoCJ Dataset	Not Found	Not Found
Simulation tool	iFogSim/ CloudSim	CloudSim	CloudSim	CloudSim
No. of Cloud hosts	4, 8, 10	4	Not Found	Not Found
No of tasks	100 to 900	500,1000, 1500, 2000	10 to 200	1000
Evaluation Metrics used	Execution time, Cost, Response Time	Makespan, AVG Execution Time, Workload Deviation	Makespan, Energy Consumption, Resource Utilization	Makespan , Throughput, Execution Time
No. Virtual Machines(VM)	70, 80, 100	30, 50	Not Found	100
Better Performing Scheduling Algorithm	GA-IRACE	K-Means clustering	ACO-BF	HSO

Ref. [49] implemented Krill, Cuckoo, and Entropy -Krill Herd Opt.(EKHO) task scheduling algorithms, used IMEAD, SPEA2 and NSGA-II datasets, CloudSim toolkit, tasks (100, 256, 512, 1000), VMs (5, 8, 10, 15, used evaluation metrics (Load Bal. Time Throughput, AVG. Turnaround Time, Latency, AVG Waiting Time, Makespan), and the authors' findings reveal that EKHO Scored the top performance, despite not revealing the number of hosts used. Ref. [50] implemented UDL, RR, PSO and HDDL task scheduling algorithms, used Python & TensorFlow toolkit, tasks (100 for testing & 1000 for training), used evaluation metrics (Energy Consumption, Latency, Cost), and the authors' findings reveal that UDL Scored the top performance, despite not revealing the name of datasets, number of hosts, and VMs. Ref. [51] implemented UMA, CEVP, EDS, and EDS-ETVMC task scheduling algorithms, used, CloudSim toolkit, 3 number of hosts, tasks (0 to 100), used evaluation metrics (Energy Consumption Response Time, Resource Utilization), and the authors' findings reveal that EDS Scored the top performance, despite not revealing the name of datasets. Ref. [52] implemented GWO, GA, FCFS, PSO, and WOA task scheduling algorithms, used HSCP, Synthetic, and GoCJ Datasets, CloudSim toolkit, 5 number of hosts, tasks (100 to 1200), 25 VMs, used evaluation metrics (Makespan, Degree of Imbalance, Throughput, AVG Resource Utilization), and the authors' findings reveal that GWO Scored the top performance, see Table 5.

Table 5. Guideline for Research Gaps Identification.

Evaluation Criteria	**Ref. [49], 2021**	**Ref. [50], 2023**	**Ref. [51], 2021**	**Ref. [52], 2021**
Task scheduling algorithm used	Krill, Cuckoo, Entropy -Krill Herd Opt.(EKHO)	UDL, RR, PSO and HDDL	UMA , CEVP, EDS, EDS-ETVMC	GWO, GA, FCFS , PSO, & WOA
Name of used dataset	IMEAD, SPEA2 and NSGA-II	Not Found	Not Found	HSCP, Synthetic, & GoCJ Datasets
Simulation tool	CloudSim	Python & TensorFlow	CloudSim	CloudSim
No. of Cloud hosts	Not Found	Not Found	3	5
No of tasks	100, 256, 512, 1000	100 for testing & 1000 for training	0 to 100	100 to 1200
Evaluation Metrics used	Load Bal. Time Throughput, AVG. Turnaround Time, Latency, AVG Waiting Time, Makespan	Energy Consumption, Latency, Cost	Energy Consumption Response Time, Resource Utilization,	Makespan, Degree of Imbalance, Throughput, AVG Resource Utilization
No. Virtual Machines (VM)	5, 8, 10, 15	Not Found	1 to 8	25
No. of Data Centers	EKHO	UDL	EDS	GWO

3. Materials and Methods

3.1. Experimental Setups

The review results show that most (52.78%) studies (19 of 36) did not reveal the name of the Datasets used including Sources. 50% (18 of 36) of studies did not reveal the number of Cloud hosts. 19.4% (7 of 36) studies did not reveal the number of Virtual Machines(VMs). All these are against replication of the research results. In light of addressing these issues, the experimental settings of the proposed approach are presented in Table 6 as follows.

Table 6. Experimental Parameter Setting.

Parameter Name	**Values**
Dataset Name	GoCJ
#of Datacenter	02
#of Task	100 to 1200
#of VM	25
#of Cloud Hosts	05
#of VM per Host	05
Policy Type of VM	Time Shared

Size of VM in MB	10,240
RAM per VM in MB	5024
Bandwidth	1000
VM MIPS	100-1200
Simulation Toolkit	CloudSim
System Configuration Details	
Processor	Intel(R) Core(TM) i5-4200M CPU @ 2.50GHz
RAM	8.00 GB
Operating system	Windows 10 Pro
System type	64-bit OS, x64-based processor

3.2. Datasets Used

The review results show that most (52.78%) studies (19 of 36) did not reveal the name of Datasets used including Sources. The top three datasets used in the reviewed studies respectively are: GoCJ dataset appeared in 9 studies, Synthetic Dataset appeared in 7 studies, and HCSP dataset appeared in 5 studies. Other used datasets include NASA dataset used in 2 studies and PlanetLab dataset used in 1 study.

As indicated in [22] the GOCJ dataset is regarded as a realistic Google dataset and was created using the workload patterns found in Google Cluster traces. For this dataset construction, a popular simulation technique known as Monte Carlo simulation is used. The GOCJ dataset contains jobs ranging in size from 15,000 to 900,000 Million Instructions (MIs). All of the tasks in this dataset fall into one of four categories: small, medium, large, extra-large, or massive. These categories range from 15k to 55k MIs, 59k to 99k MIs, 101k to 135k MIs, 150k to 337.5k MIs, and 525k to 900k MIs, respectively [22]. Our study used GoCJ (Google Cloud Jobs) dataset to conduct experiments considering its popularity, being recent, and having dataset of varied sizes (from small to very large). Moreover, it reflects actual workload behavior in the cluster traces of Google and helpfulness for researchers' community working on task scheduling, load balancing scheduler, and other cloud-based applications using realistic HPC (High-Performance Computing) jobs in the setting of cloud computing, details the dataset can be accessed in Ref. [53].

As illustrated in Table 7, 100 cloudlets/tasks have 19 small workloads with a range of MIPS between 15K and 55K, 39 medium workloads with a range of MIPS between 59K and 99K, 31 large workloads with a range of MIPS between 101K and 135K, 3 extra-Large workloads with a range of MIPS between 150K and 337.5K, 8 huge workloads with a range of MIPS between 525K and 900K. 200 to 700 cloudlets/tasks also have their respective small to huge workloads and MIPS range. The details of workload type and MIPS are presented in [52,53].

Table 7. List of Workload type and Cloudlets/tasks with its MIPSWorkload type.

	MIPS	**Cloudlets/Tasks**						
		100	200	300	400	500	600	700
Small	15K–55K	19	42	61	55	96	114	116
Medium	59K–99K	39	75	114	165	191	229	295
Large	101K–135K	31	63	95	139	155	190	222
Extra large	150K–337.5K	3	4	7	18	25	25	29
Huge	525K–900K	8	16	23	23	33	42	38

3.3. Task (Job) Scheduling Algorithms Selection Methods

As illustrated in Table 1, diverse task scheduling algorithms were used by the scientific community despite not all of them performing at the top level in this regard. The study screened out the top-performing task scheduling algorithms from different studies to enable upcoming researchers to concentrate more on these algorithms to enhance Cloud computing performance while not ignoring the rest. The better-performing task scheduling algorithms across 36 reviewed studies include ANN-BPSO. Q-Learning, GA, DRALB, Min-Min, EMVO, ANN–FOA, H3CSA, Tournament Selection-GA i.e. TS-GA, DAIRS with Q-Learning, ACO, ABC-Q-Learning, GA-ACO, GWO, GA-IRACE, K-Means clustering, ACO-BF, HSO, EKHO, UDL, EDS, and MHDNNL.

Relative analysis results show that GA algorithm and its variants, ACO, and Q-Learning, were among those commonly appearing scheduling algorithms in different studies as the top performers on individual bases, as well as or when applied with other task scheduling algorithms. In this regard, the proposed approach employed the GA algorithm along with the EDF algorithm (GA-EDF) on a hybrid

basis, as none of the examined research used Earliest Deadline First (EDF). EDF scheduler is popular and performs best in situations where jobs' energy can be used greedily. If a minimum of one legitimate task set schedule exists, then any set of periodic tasks along with deadlines are successfully scheduled by EDF. This is known as optimality [54]. The EDF is keeping all of the jobs in a queue that are prepared for execution. The end of the queue would be where any newly arrived tasks would be placed. Task selection (to run next) and deletion would take O (n) time, where n is the number of tasks in the queue. Each task insertion will be completed in O (1), or constant time. EDF just uses a heap data structure to keep all available tasks in a sorted priority queue. A record for a job can be added to the heap in O (log2 n) time upon its arrival, where n is the total number of tasks in the priority queue [3].

The flowchart of the proposed dynamic task scheduling algorithm is presented in Figure 1 as follows. The flow chart starts with submitting tasks, and ends ups to assigning tasks to the VM. First, the assignments that have been submitted are arranged according to their cost and deadline. Task costs are categorized as high, medium, and low before a selection of tasks by GA-EDF. Tasks are prioritized according to high, medium, and low priority before being put in ascending order for the deadline by GA-EDF. Selected tasks based on cost and deadline are then merged, GA-EDF verifies the state of VMs to distribute tasks based on current load, and at last, GA-EDF assigns tasks to the appropriate VMs. The flowchart of the GWO algorithm is illustrated in the study [52].

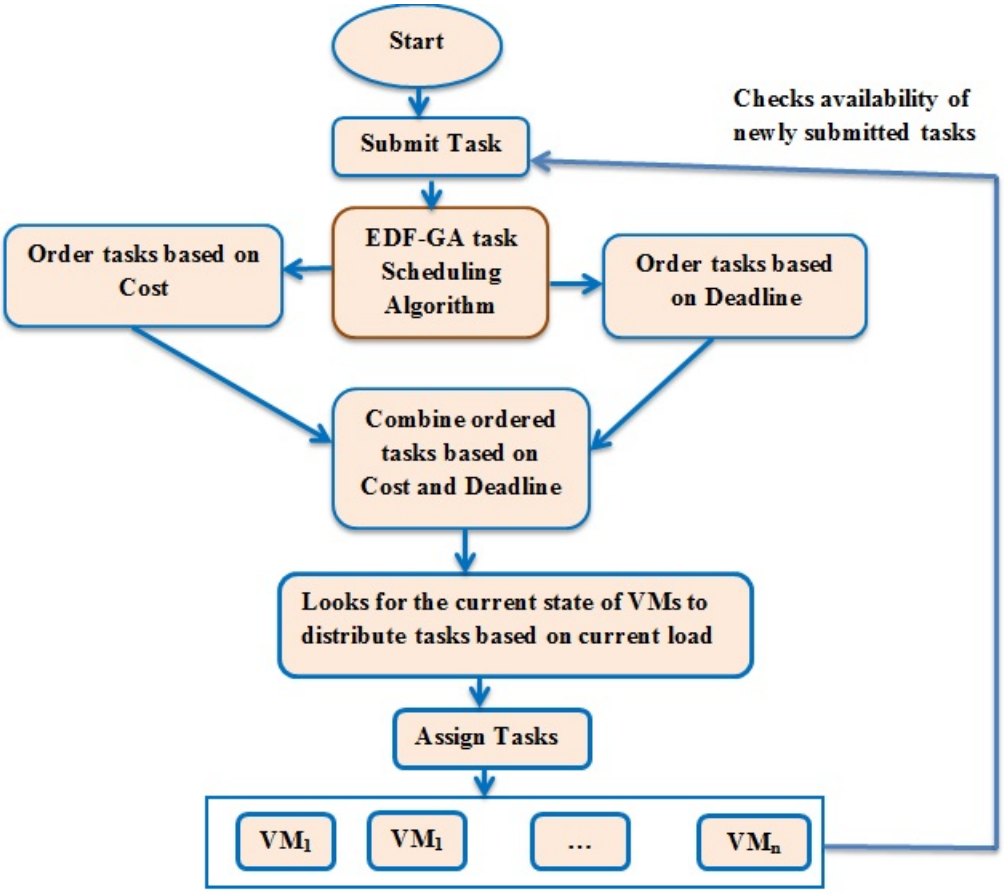

Figure 1. Flowchart of EDF-GA task scheduling algorithms.

3.4. Simulation Toolkit Used

The top simulation toolkits used in the reviewed studies respectively are: CloudSim appeared in 26 of 36 studies, MATLAB appeared in 3 studies, Python & TensorFlow appeared in 2 studies, GCC compiler V.4.4.4 and ifogsim appeared in 1 study. This finding reveals that CloudSim is the top preferred toolkit for task scheduling experiments used in 72.22% of studies. These are all the core rationales for choosing the CloudSim toolkit for the proposed study.

3.5. Evaluation Metrics

The top performance evaluation metrics in the reviewed studies respectively are: Makespan appeared

in 22 of 36 studies, Resource Utilization appeared in 18 of 36 studies; Throughput appeared in 13 of 36 studies. Considering their popularity, the study employed Makespan, resource utilization, and throughput evaluation metrics to evaluate the proposed task scheduling algorithms i.e. GA-EDF, and compared the results of the proposed approach with another study, which employed the same datasets and a comparable experimental setup. Brief details and formulas of the top three evaluation metrics used in this study are presented as follows and it can be accessed from Ref. [22,27,52].

3.6. Makespan

The least Makespan results are a good sign for assigning the tasks/jobs to Virtual Machines (VMs) with lower waiting and execution/completion times. Makespan calculation is presented in Eq. (1):

$$\text{Makespan} = \text{Max} = \text{Max}\{\text{CT1}, \text{CT2}, \ldots, \text{CTn}\} \tag{1}$$

where n signify number of Virtual Machines (VMs) and CT signify Virtual Machines (VMs) completion time VM_i.

3.7. Resource Utilization on Average

An efficient scheduling algorithm can be determined by higher resource utilization values in which all tasks/jobs are completely executed on the existing Virtual Machines (VMs). Average Utilization of Resources (AUR) is calculated in Eq. (2).

$$\text{AUR} = \frac{\text{MakespanAVG}}{\text{Makespan}} \tag{2}$$

where MakespanAVG is computed in Eq. (3):

$$\text{MakespanAVG} = \frac{\sum_{i=1}^{n} \text{Makespan}i}{n} \tag{3}$$

3.*8. Throughput*

An efficient scheduling algorithm can be determined by higher throughput values in which the total number of tasks/jobs being completed/ executed per second or per unit time. Throughput is computed in Eq. (4):

$$\text{Throughput} = \frac{\text{TotalJobs i. e. number of jobs}}{\text{Makespan}} \tag{4}$$

4. Result and Discussions

Nine (9 out of 36) different studies used a GoCJ dataset like ours despite some did not reveal the number of Cloud hosts and VMs, simulation toolkit used, did not consider evaluation metrics like throughput, Makespan, and resource utilization and none of them apply GA-EDF scheduling algorithms.

Relative analysis results show that GA task scheduling algorithm and its variants, ACO, and Q-Learning, were among those commonly appearing scheduling algorithms in different studies as the top performers on individual bases, as well as or when applied with other task scheduling algorithms. According to [3] real-time systems are ones whose proper functioning is contingent upon both the timing of the logical outcomes as well as their production. These are extremely sophisticated systems that are used in a variety of settings, including distributed systems, robotics, aviation, multimedia, and military process control. In order to ensure that no task (for hard real-time systems) or a minimum number of tasks (for soft real-time systems) miss their deadlines, scheduling algorithms are used in real-time systems to determine the sequence in which tasks are executed and the amount of time allotted for each task. In this regard, the viability of the EDF algorithm in online or real time task scheduling was indicated in [3] despite not considering GA task scheduling algorithms for faster convergence in big scheduling problems and is efficient in reducing energy usage, computational cost, and makespan, as well as effective in avoiding local optima [1].

Given the advantages of algorithms, the fact that GA is popular in the majority of evaluated studies, and the fact that none of the 36 papers that were investigated used Earliest Deadline First (EDF), the study used the GA algorithm in conjunction with the EDF algorithm (GA-EDF) on a hybrid basis. According to recent study [2], compared to other meta-heuristic algorithms, GWO has recently become popular because of its many advantages, including ease in implementation, fewer parameters that result in minimal complexity time and energy usage, and convergence during execution. Numerous common

meta-heuristics are used for task scheduling. However, these algorithms have multiple search methods, randomness issues, limited global search capabilities, and less convergence in the late iteration, which prevents them from finding the local optimum search solution. Additionally, there is an imbalance between local and global search. The majority of these issues are resolved by the grey wolf optimizer (GWO). Compared to the PSO technique, the GWO mechanism uses less memory because it only needs the position of one vector. Furthermore, unlike the PSO algorithm, which selects one best solution by all particles, GWO depends on the three best solutions to prevent it from falling into the local optimal solution. Furthermore, compared to GA algorithms, the GWO algorithm has fewer parameters, meaning it is simpler and uses less energy and processing time. Compared to Ant Colony optimization, which ignores the dynamic aspect of computing resources, the GWO algorithm is more dynamic [2]. As compared to PSO and GA, faster convergence was attained by GWO algorithm in the recent study [22]. These are one of the core rationale for including the EDF-GA, GWO and GA scheduling algorithms in our study.

The study aimed to identify the task scheduling algorithm that maximizes resource utilization and throughput while reducing Makespan. The results of another study [22,52], which employed the same datasets and a comparable experimental setup to evaluate the GA and GWO task scheduling algorithms using the Makespan metric compared with the results of the proposed approach (GA-EDF). Compared to the proposed approach (GA-EDF combination) and GWO algorithm, the GA task scheduling algorithms demonstrate poor performance based on Makespan, average resource utilization, and throughput.

Based on the Makespan evaluation metric, the proposed hybrid algorithm (GA-EDF combination) attained top performance across different workloads (100 to 1000) in terms of reducing Makespan, while GWO and GA task scheduling algorithms ranked second and third, respectively, in the aforementioned workloads (see Figure 2). The experimental findings show that the average Makespan reduction of the proposed approach is 6.16% larger than that of the GWO algorithm and 24.99% higher than the GA based on the results presented in Table 8. In a recent study [22], the GWO algorithm outperformed the GA and PSO task scheduling algorithms in terms of Makespan reduction on similar datasets (GoCJ dataset) and workloads 100 to 1200, despite the study's failure to conduct experiments on the GA-EDF combination. In the study [52] the GWO outperformed PSO, GA, and WOA task scheduling algorithms in Makespan reduction on a similar dataset (GoCJ dataset) despite the study's failure to conduct experiments on GA-EDF combination. In our study, combining GA with EDF shows relatively superior performance across different workloads in minimizing Makespan when compared to the solo performance of GA and GWO task scheduling algorithms.

Table 8. Experiment Results on Makespan.

Cloudlets/ Tasks	**GA Algorithm**	**GWO Algorithm**	**Proposed Approach GA-EDF Algorithm**	**Improved Makespan by the Proposed Approach**		**Makespan from Job (Small, Medium, Large, Extra-Large, and Huge Jobs Respectively**
				Against GA	**Against GWO**	
100	678	420	375	303	45	19,39,31,3, & 8
200	978	713	660	318	53	42,75,63,4, & 16
300	1278	1006	945	333	61	61,114,95,7, & 23
400	1578	1299	1230	348	69	55,165,139,18, & 23
500	1878	1592	1515	363	77	96,191,155,25, & 33
600	2178	1885	1800	378	85	114,229,190,25, & 42
700	2478	2178	2085	393	93	116,295,222,29 &38

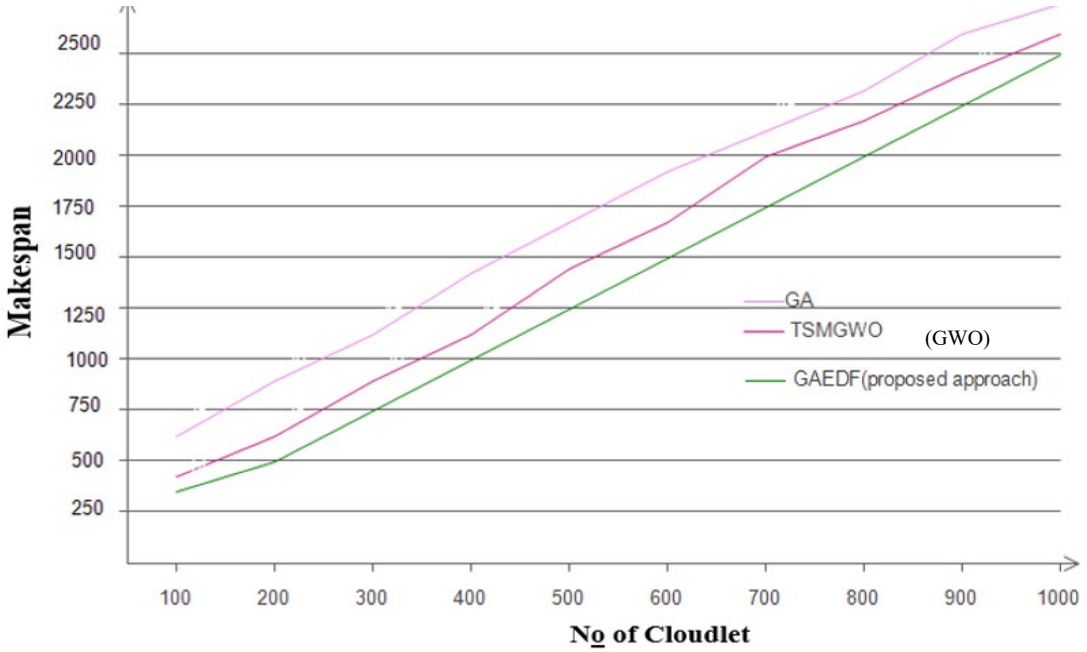

Figure 2. Experiment Results on Makespan.

The results of another study [52] employed the same datasets and a comparable experimental setup to evaluate the GA and GWO task scheduling algorithms using the Average Resource Utilization metric compared with the results of the proposed approach (GA-EDF combination). Even though the GWO algorithm performed slightly better than the proposed approach on workloads 300 and 500, the GA-EDF algorithm's average resource utilization is 63.28% higher than the GA average resource utilization and 1.23% higher than the GWO algorithm based on the results presented in Table 9. The proposed approach Average Resource Utilization over GA is 0.21 on workload 100, 0.26 on workload 200, 0.09 on workload 300, 0.27 on workload 400, 0.24 on workload 500, 0.46 on workload 600, and 0.31 on workload 700. The proposed approach Average Resource Utilization over GWO is 0.01 on workload 100, 0.02 on workload 200, 0.03 on workload 400, 0.02 on workload 600, and 0.01 on workload 700. The Average Resource Utilization achieved by the GWO Algorithm is better than the proposed approach by 0.01 on workload 300 and 0.02 on workload 500. The Average Resource Utilization of the proposed approach is relatively better than the GA and GWO algorithms on workloads 800, 900, and 1000 as illustrated in Figure 3.

Table 9. Experiment Results on Average Resource Utilization.

Cloudlets/ Tasks	**GA Algorithm**	**TSMGWO Algorithm**	**Proposed Approach GA-EDF Algorithm**	**Improved Makespan by the Proposed Approach**		**Average Resource Utilization from Jobs (small, Medium, Large, Extra Large, and Huge Jobs, Respectively**
				Against GA	**Against GWO**	
100	0.25	0.45	0.46	0.21	0.01	19,39,31,3, & 8
200	0.35	0.59	0.61	0.26	0.02	42,75,63,4, & 16
300	0.4	0.5	0.49	0.09	-0.01	61,114,95,7, & 23
400	0.51	0.75	0.78	0.27	0.03	55,165,139,18, & 23
500	0.55	0.81	0.79	0.24	–0.02	96,191,155,25, & 33
600	0.41	0.85	0.87	0.46	0.02	114,229,190,25, & 42
700	0.58	0.88	0.89	0.31	0.01	116,295,222,29 &38

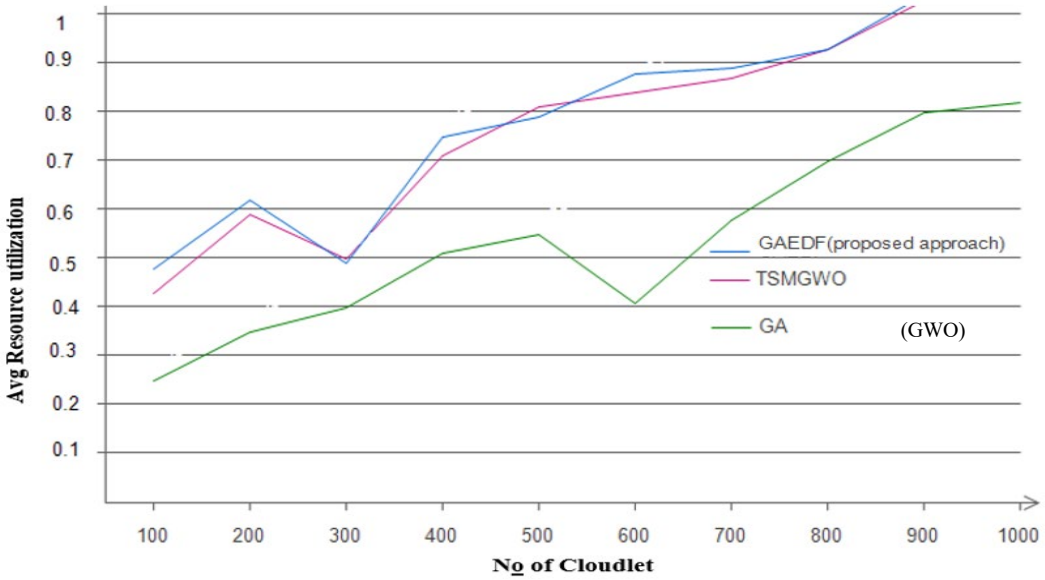

Figure 3. Experiments on Average Resource Utilization.

According to the throughput evaluation metric, the proposed hybrid approach (GA-EDF combination) outperformed the solo performance of GA in terms of increasing throughput across various workloads (100, 200, 300, 400, and 500) with average scores of 0.32, 0.2, 0.31, 0.1, and 0.04, respectively. This was true even though GA performance increased on workloads 600 and 700 with average scores of 0.02 and 0.05, respectively, and declined on workloads 800, 900, and 1000 compared to the proposed approach (see Figure 4). This finding indicates the proposed approach is relatively better at increasing throughput in most small workloads and larger workloads when compared with GA. Even though the GWO algorithm performed equally well on workload 400 and lower performance on workloads 100, 200, and 300, it performed better than the proposed approach on workloads 500, 600, 700, 800, 900, and 1000 as illustrated in Figure 2. Despite the proposed approach performance inconsistency across different workloads; the average throughput increment of the proposed approach is 43.23% higher than the GA average throughput and 8.32% higher than the GWO algorithm average throughput based on the results presented in Table 10. In a recent study [22], the GWO algorithm outperformed the GA and PSO task scheduling algorithms in increasing throughput on similar datasets (GoCJ dataset) and workloads 100 to 1200, despite the study's failure to conduct experiments on the GA-EDF combination. In the study [52] the GWO outperformed PSO, GA, and WOA task scheduling algorithms in increasing throughput on a similar dataset (GoCJ dataset) despite the study's failure to conduct experiments on GA-EDF combination.

Table 10. Experiment Results on Throughput.

Cloudlets/Tasks	**GA Algorithm**	**GWO Algorithm**	**Proposed Approach GA-EDF Algorithm**		**Improved Makespan by the Proposed Approach**	**Throughput from Jobs (Small, Medium, Large, Extra Large, and Huge Jobs, Respectively**
				GA	GWO	
100	0.25	0.32	0.57	0.32	0.25	19,39,31,3, & 8
200	0.3	0.36	0.50	0.2	0.14	32,75,63,4, & 16
300	0.33	0.39	0.64	0.31	0.25	61,114,95,7, & 23
400	0.35	0.45	0.45	0.1	0	55,165,139,18, & 23
500	0.38	0.49	0.42	0.04	−0.07	96,191,155,25, & 33
600	0.4	0.53	0.38	−0.02	−0.15	114,229,190,25, & 42
700	0.43	0.58	0.38	−0.05	−0.2	116,295,222,29 &38

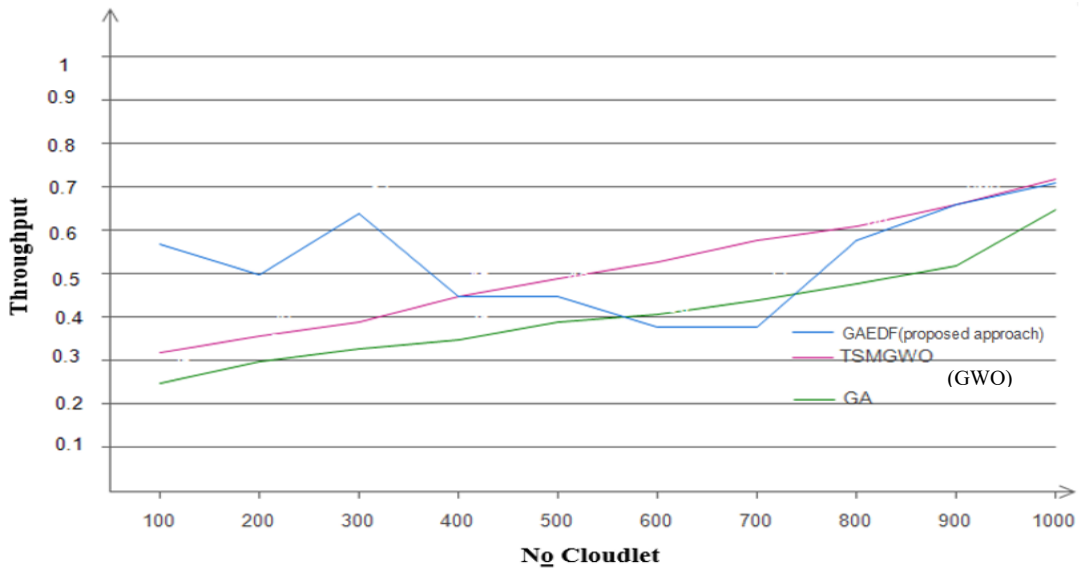

Figure 4. Experiment Results on Throughput.

5. Limitation of the Study

Although the study 1) used structured guidelines to excavate open issues and best current practices regarding different task scheduling algorithms in cloud computing field from recent and pertinent research articles extracted from SCOPUS and WoS Databases, published between 2020 and 2024, 2) explored the type of task scheduling algorithms utilized, the dataset source, the number of tasks/workload, the number of cloud hosts, the simulation toolkits, the kind of evaluation metrics used, the number of virtual machines, and the scheduling techniques that performed best overall; 3) employed the frequently top-performing task scheduling algorithm in the different reviewed studies like Genetic Algorithm (GA) with task scheduling algorithms that is not incorporated in the reviewed studies the Earliest Deadline First (EDF) as novel practical contributions. The study has the following major limitations.

The proposed approach's reliability has not been examined in real-time settings or on a variety of datasets. This implies that when used in a real-world environment, the experimental results produced by Cloudsim might not be consistent. Future research is expected to include over 1000 workloads/the number of tasks.

The study includes a few dynamic task-scheduling algorithms from the reviewed literature considering their intrinsic properties in optimizing the task-scheduling process specifically in reducing high computational complexity, slow convergence, and inefficiency in dynamic environments. It is done due to the fact that GA is a popular and highly effective task scheduling algorithm in a number of reviewed studies, that EDF is suitable for real-time (online) scheduling, that higher-priority jobs are executed on faster processors in EDF [3], and that none of the 36 reviewed studies used the EDF task scheduling algorithm, the study proposed hybrid task scheduling algorithms that combine GA and EDF to fill the literature gaps. Recent developments in task scheduling algorithms are expected to be included in future research, such as exploring the potential of machine learning, deep learning, and ensemble learning-based approaches.

The review finding reveals that CloudSim is the top preferred toolkit for task scheduling experiments used in 72.22% of studies (appeared in 26 of 36 studies). These are all the core rationales for choosing the CloudSim toolkit for the proposed study. Future research is expected to conduct experiments on fog-server computing environment to compare the experimental results on Cloudsim because Fog computing is a potent extension of cloud computing and services to the network's edge, has emerged in recent years. Fog offers end users data, computation, storage, and application services, much as Cloud. By bringing basic computing and analytical services to the network's edge, bringing computing resources closer to end devices, facilitating mobility, enhancing overall network performance and efficiency, and lowering latency, fog computing helps to lessen the workload associated with cloud computing [45,55,56].

Considering the wider acceptance and reliability, the extractions of reviewed research works are limited to Scopus and Web of Science databases and open-access publications from 2020 to 2024.

6. Conclusion

Time-dependent sectors like banks, healthcare, airlines, online shopping, and more suffer from slow delivery of cloud computing services unless intervention of a fast, reliable, and efficient task-scheduling algorithm is made to reduce computing resource consumption while enhancing the cloud computing customer satisfaction. In this regard, the study starts with conducting rigorous assessment of recent and pertinent research articles extracted from SCOPUS and WoS Databases, open-access publications from 2020 to September 2024, used structured guidelines to excavate open issues and best current practices regarding different task scheduling algorithms, explore the type of task scheduling algorithms utilized, the dataset source, the number of tasks given, the number of cloud hosts, the simulation toolkits, the kind of evaluation metrics used, the number of virtual machines, and the scheduling techniques that performed best overall in 36 systematically reviewed studies. It compared the proposed approach's performances against the counterpart in popular evaluation metrics like resource utilization, Makespan, and throughput.

Review findings show that more studies fail to reveal experimental settings like dataset source, number of hosts, number of VMs, and simulation toolkits which is against replication of results. The top performance evaluation metrics in the reviewed studies respectively are: Makespan appeared in 22 of 36 studies, Resource Utilization appeared in 18 of 36 studies, Throughput appeared in 13 of 36 studies, Response Time appeared in 13 of 36 studies., Energy Consumption appeared in 7 of 36 studies, Cost appeared in 6 of 36 studies, and latency appeared in 3 studies. The top simulation toolkits used in the reviewed studies respectively are: CloudSim appeared in 26 of 36 studies, MATLAB appeared in 3 studies, Python & TensorFlow appeared in 2 studies, and GCC compiler V.4.4.4 appeared in 1 study. The top three datasets used in the reviewed studies respectively are: The GoCJ dataset appeared in 9 studies, the Synthetic Dataset appeared in 7 studies, and the HCSP dataset appeared in 5 studies. Other used datasets include the NASA dataset used in 2 studies and the PlanetLab dataset used in 1 study. The proposed study selected a dataset, simulation toolkit, and evaluation metrics considering their popularity in the reviewed studies.

The study applied one of the top performing task scheduling algorithms in the reviewed studies i.e. Genetic Algorithm (GA) along with a task scheduling algorithm used in none of the reviewed research works i.e. Earliest Deadline First (EDF) considering its suitability for the real time or online scheduling.

The overall experiment results show that the proposed task scheduling algorithm is relatively better in maximizing average resource utilization and throughput while reducing Makespan despite slightly less performing than the GWO algorithm on workload 500 for average resource utilization, slightly less performing than the GWO algorithm on workloads 500, 600, 700, 800, 900, and 1000 in maximizing throughput, and slightly less performing than GA algorithm on workloads 500 and 600 in maximizing throughput. These experimental results demonstrate that the proposed approach outperforms GA and GWO algorithms in terms of minimizing Makespan across all workloads and optimizing resource utilization on all workloads, except on workloads 300 and 500, where it performs worse than GWO algorithms. Compared to the proposed approach (GA-EDF combination) and GWO algorithm, the GA task scheduling algorithms demonstrate poor performance based on Makespan, average resource utilization, and throughput.

The results of the proposed approach are not examined in real-time settings or on multiple datasets. This implies that when used in a real-world environment, the experimental results produced by Cloudsim might not be consistent. Aligning with the proposed approach, future research is expected to explore the potential of machine learning, deep learning, and ensemble learning-based approaches in optimizing the process of task scheduling on Cloudsim, fog server computing, and real environments.

Author Contributions

Conceptualization, K.A., K.M. and K.K.; Methodology, K.A., K.M., and K.K.; Software, K.M.; Validation, K.A. and K.M.; Formal analysis, K.A. and K.M.; Investigation, K.A, K.M., and K.K.; Resources, K.M.; Data Curation, K.A., K.M. and K.K.; Writing—original draft preparation, K.A.; Writing—review and editing, K.A., K.M. and K.K.; Visualization, K.A. and K.M.; Supervision, K.A. and K.K.; Project administration, K.A., K.M. and K.K. All authors have read and agreed to the published version of the manuscript.

Funding

This research received no external funding.

Conflict of Interest Statement

The authors declare no conflicts of interest.

Data Availability Statement
Not applicable.

Acknowledgments
The authors thank all participating editors and anonymous reviewers for their valuable suggestions and comments to make the manuscript up to standard.

References

1. Behera et al. "Task scheduling optimization in heterogeneous cloud computing environments: a hybrid GA-GWO approach, "*J. Parallel Distrib. Comput.*, Volume 183, 2024, doi: https://doi.org/10.1016/j.jpdc.2023.104766
2. F. A. Saif, R. Latip, Z. M. Hanapi, and K. Shafinah, "Multi-Objective Grey Wolf Optimizer Algorithm for Task Scheduling in Cloud-Fog Computing," *IEEE Access*, vol. 11, no. December 2022, pp. 20635–20646, 2023, doi: 10.1109/ACCESS.2023.3241240.
3. J. Singh, B. Patra, and S. Prasad, "An Algorithm to Reduce the Time Complexity of Earliest Deadline First Scheduling Algorithm in Real-Time System," *Int. J. Adv. Comput. Sci. Appl.*, vol. 2, no. 2, 2011, doi: 10.14569/ijacsa.2011.020207.
4. R. Arunarani, D. Manjula, and V. Sugumaran, "Task scheduling techniques in cloud computing: A literature survey," Future Generation Computer Systems, vol. 91. pp. 407–415, 2019. doi: 10.1016/j.future.2018.09.014.
5. L. Yin, J. Luo, and H. Luo, "Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing," *IEEE Trans. Ind. Informatics,* vol. 14, no. 10, pp. 4712–4721, 2018,doi:10.1109/TII.2018.2851241
6. S. Basu et al., "Cloud computing security challenges & solutions-A survey," in *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, IEEE, 2018, pp. 347–356, doi: 10.1109/CCWC.2018.8301700
7. J. Blazewicz, K. Ecker, E. Pesch, G. Schmidt, and J. Weglarz, *Handbook on scheduling: From Theory to Practice*, Springer, 2019, doi: https://doi.org/10.1007/978-3-319-99849-7
8. E. H. Houssein, A. G. Gad, Y. M. Wazery, and P. N. Suganthan, "Task Scheduling in Cloud Computing based on Meta-heuristics: Review, Taxonomy, Open Challenges, and Future Trends," *Swarm Evol. Comput.*, vol. 62, p. 100841, 2021, doi: https://doi.org/10.1016/j.swevo.2021.100841.
9. N. Soltani, B. Soleimani, and B. Barekatain, "Heuristic algorithms for task scheduling in cloud computing: a survey," *Int. J. Comput. Netw. Inf. Secur.*, vol. 11, no. 8, p. 16, 2017, doi: https://doi.org/10.5815/ijcnis.2017.08.03
10. P. Zhang and M. Zhou, "Dynamic cloud task scheduling based on a two-stage strategy," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 2, pp. 772–783, 2017, doi: 10.1109/TASE.2017.2693688
11. V. M. Arul Xavier and S. Annadurai, "Chaotic social spider algorithm for load balance aware task scheduling in cloud computing," *Cluster Comput.*, vol. 22, no. Suppl 1, pp. 287–297, 2019, doi: https://doi.org/10.1007/s10586-018-1823-x
12. X. Yang and N. Rahmani, "Task scheduling mechanisms in fog computing: review, trends, and perspectives," *Kybernetes*, vol. 50, no. 1, pp. 22–38, 2021, doi: https://doi.org/10.1108/K-10-2019-0666
13. J. Zhou et al., "Fault-tolerant task scheduling for mixed-criticality real-time systems," *J. Circuits, Syst. Comput.*, vol. 26, no. 01, p. 1750016, 2017,doi: https://doi.org/10.1142/S0218126617500165
14. T. Aladwani, "Types of task scheduling algorithms in cloud computing environment," *Sched. Probl. Appl. Trends*, pp. 1–12, 2020, doi: 10.5772/intechopen.86873
15. M. Agarwal and G. M. S. Srivastava, "A PSO algorithm based task scheduling in cloud computing," *Int. J. Appl. Metaheuristic Comput.*, vol. 10, no. 4, pp. 1–17, 2019, doi: https://doi.org/10.1007/978-981-13-0589-4_27
16. X. Wei, "Task scheduling optimization strategy using improved ant colony optimization algorithm in cloud computing," *J. Ambient Intell. Humaniz. Comput.*, pp. 1–12, 2020,doi: https://doi.org/10.1007/s12652-020-02614-7
17. M. I. Alghamdi, "Optimization of Load Balancing and Task Scheduling in Cloud Computing Environments Using Artificial Neural Networks-Based Binary Particle Swarm Optimization (BPSO)," *Sustainability* (Switzerland), vol. 14, no. 19. 2022. doi: 10.3390/su141911982.
18. D. Ding, X. Fan, Y. Zhao, K. Kang, Q. Yin, and J. Zeng, "Q-learning based dynamic task scheduling for energy-efficient cloud computing," *Future Generation Computer Systems*, vol. 108. pp. 361–371, 2020. doi: 10.1016/j.future.2020.02.018.
19. A. A. Gad-Elrab, T. A. Alzohairy, K. R. Raslan, and F. A. Emara, "Genetic-Based Task Scheduling Algorithm with Dynamic Virtual Machine Generation in Cloud Computing," *Int. J. Comput.*, vol. 20, no. 2, pp. 165–174, 2021, doi: https://doi.org/10.47839/ijc.20.2.2163
20. S. Chhabra and A. K. Singh, "Dynamic resource allocation method for load balance scheduling over cloud data center networks," J. Web Eng., vol. 20, no. 8, pp. 2269–2284, 2021, doi: 10.13052/jwe1540-9589.2083
21. R. Kaur, V. Laxmi, and Balkrishan, "Performance evaluation of task scheduling algorithms in virtual cloud environment to minimize makespan," *Int. J. Inf. Technol.*, vol. 14, no. 1, pp. 79–93, 2022, doi: 10.1007/s41870-021-00753-4.
22. K. Sreenu and S. Malempati, "Multiple Resource Attributes and Conditional Logic Assisted Task Scheduling in Cloud Computing.," *Int. J. Intell. Eng. Syst.*, vol. 16, no. 3, 2023, doi: 10.22266/ijies2023.0630.54
23. S. E. Shukri, R. Al-Sayyed, A. Hudaib, and S. Mirjalili, "Enhanced multi-verse optimizer for task scheduling in cloud computing environments," *Expert Syst. Appl.*, vol. 168, p. 114230, 2021, doi: https://doi.org/10.1016/j.eswa.2020.114230

24. P. Memari, S. S. Mohammadi, F. Jolai, and R. Tavakkoli-Moghaddam, “A latency-aware task scheduling algorithm for allocating virtual machines in a cost-effective and time-sensitive fog-cloud architecture,” *J. Supercomput.*, vol. 78, no. 1, pp. 93–122, 2022, doi: https://doi.org/10.1007/s11227-021-03868-4
25. F. Alhaidari and T. Z. Balharith, “Enhanced round-robin algorithm in the cloud computing environment for optimal task scheduling,” *Computers*, vol. 10, no. 5, p. 63, 2021, doi: https://doi.org/10.3390/computers10050063
26. T. Bezdan, M. Zivkovic, N. Bacanin, I. Strumberger, E. Tuba, and M. Tuba, “Multi-objective task scheduling in cloud computing environment by hybridized bat algorithm,” *J. Intell. Fuzzy Syst.*, vol. 42, no. 1, pp. 411–423, 2022,doi: https://doi.org/10.3233/JIFS-2192
27. S. Nabi, M. Ibrahim, and J. M. Jimenez, “DRALBA: Dynamic and Resource Aware Load Balanced Scheduling Approach for Cloud Computing,” *IEEE Access*, 2021, doi: 10.1109/ACCESS.2021.3074145.
28. G. Sharma, N. Miglani, and A. Kumar, “PLB: a resilient and adaptive task scheduling scheme based on multi-queues for cloud environment,” *Cluster Comput.*, 2021, doi: 10.1007/s10586-021-03280-w.
29. Mishra, M. Narayan Sahoo, and A. Satpathy, “H3CSA: A makespan aware task scheduling technique for cloud environments,” *Trans. Emerg. Telecommun. Technol.*, 2021, doi: 10.1002/ett.4277.
30. Z. Peng, P. Pirozmand, M. Motevalli, and A. Esmaeili, “Genetic algorithm-based task scheduling in cloud computing using mapreduce framework,” *Math. Probl. Eng.*, vol. 2022, 2022,doi: https://doi.org/10.1155/2022/4290382
31. Y. Hamed and M. H. Alkinani, “Task scheduling optimization in cloud computing based on genetic algorithms,” *Comput. Mater. Contin*, vol. 69, no. 03, pp. 3289–3301, 2021,doi: https://doi.org/10.32604/cmc.2021.018658
32. S. Sawwashere, “Improvement in Task Scheduling Capabilities for SaaS Cloud Deployments Using Intelligent Schedulers,” *Int. J. Big Data Anal. Healthc.*, 2021, doi: 10.4018/ijbdah.287104.
33. Q. Li, Z. Peng, D. Cui, J. Lin, and J. He, “MHDNNL: A Batch Task Optimization Scheduling Algorithm in Cloud Computing,” *Int. J. Inf. Technol. Web Eng.*, 2022, doi: 10.4018/IJITWE.310053.
34. X. Zhang, “A fine-grained task scheduling mechanism for digital economy services based on intelligent edge and cloud computing,” *J. Cloud Comput*., 2023, doi: 10.1186/s13677-023-00402-0.
35. 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,” *IEEE Access*, 2024, doi: 10.1109/ACCESS.2024.3364700
36. R. Ravinder and N. Vurukonda, “Cloud computing: an efficient load balancing and scheduling of task method using a hybrid optimization algorithm,” Indones. *J. Electr. Eng. Comput. Sci.*, 2023, doi: 10.11591/ijeecs.v32.i3.pp1545-1556.
37. X. He, J. Shen, F. Liu, B. Wang, G. Zhong, and J. Jiang, “A two-stage scheduling method for deadline-constrained task in cloud computing,” *Cluster Comput.*, 2022, doi: 10.1007/s10586-022-03561-y.
38. B. Kruekaew and W. Kimpan, “Multi-Objective Task Scheduling Optimization for Load Balancing in Cloud Computing Environment Using Hybrid Artificial Bee Colony Algorithm with Reinforcement Learning,” *IEEE Access*, 2022, doi: 10.1109/ACCESS.2022.3149955.
39. Mubeen, M. Ibrahim, N. Bibi, M. Baz, H. Hamam, and O. Cheikhrouhou, “Alts: An adaptive load balanced task scheduling approach for cloud computing,” *Processes*, 2021, doi: 10.3390/pr9091514.
40. M. S. A. Khan and R. Santhosh, “Task scheduling in cloud computing using hybrid optimization algorithm,” *Soft Comput.*, 2022, doi: 10.1007/s00500-021-06488-5.
41. M. Lavanya, B. Shanthi, and S. Saravanan, “Multi objective task scheduling algorithm based on SLA and processing time suitable for cloud environment,” *Comput. Commun.*, 2020, doi: 10.1016/j.comcom.2019.12.050.
42. R. Khorsand and M. Ramezanpour, “An energy-efficient task-scheduling algorithm based on a multi-criteria decision-making method in cloud computing,” *Int. J. Commun. Syst.*, 2020, doi: 10.1002/dac.4379.
43. R. Gulbaz, A. B. Siddiqui, N. Anjum, A. A. Alotaibi, T. Althobaiti, and N. Ramzan, “Balancer genetic algorithm-a novel task scheduling optimization approach in cloud computing,” *Appl. Sci.*, 2021, doi: 10.3390/app11146244.
44. D. Srikar* and T. Manoranjitham, “Efficient Task Scheduling for Quality of Service in Cloud Computing Network,” *Int. J. Recent Technol. Eng.,* 2020, doi: 10.35940/ijrte.e6400.018520.
45. J. U. Arshed, M. Ahmed, T. Muhammad, M. Afzal, M. Arif, and B. Bazezew, “GA-IRACE: Genetic Algorithm-Based Improved Resource Aware Cost-Efficient Scheduler for Cloud Fog Computing Environment,” *Wirel. Commun. Mob. Comput.*, 2022, doi: 10.1155/2022/6355192.
46. G. Muthusamy and S. R. Chandran, “Cluster-based task scheduling using K-means clustering for load balancing in cloud datacenters,” *J. Internet Technol.*, 2021, doi: 10.3966/160792642021012201012
47. F. U. Zambuk, A. Y. u. Gital, M. Jiya, N. A. S. Gari, B. Ja’afaru, and A. Muhammad, “Efficient Task Scheduling in Cloud Computing using Multi-objective Hybrid Ant Colony Optimization Algorithm for Energy Efficiency,” *Int. J. Adv. Comput. Sci. Appl.*, 2021, doi: 10.14569/IJACSA.2021.0120353.
48. N. Kumar, U. Dugal, and A. Singh, “Optimizing Task Scheduling in Cloud Computing Environments using Hybrid Swarm Optimization,” *J. Comput. Mech. Manag.*, 2023, doi: 10.57159/gadl.jcmm.2.5.23076.
49. B. B. Naik, D. Singh, and A. B. Samaddar, “Multi-objective Virtual Machine Selection in Cloud Data Centers Using Optimized Scheduling,” *Wirel. Pers. Commun.,* 2021, doi: 10.1007/s11277-020-07807-z.
50. Q. Li, Z. Peng, D. Cui, J. Lin, and H. Zhang, “UDL: a cloud task scheduling framework based on multiple deep neural networks,” *J. Cloud Comput.*, 2023, doi: 10.1186/s13677-023-00490-y.
51. Marahatta, S. Pirbhulal, F. Zhang, R. M. Parizi, K. K. R. Choo, and Z. Liu, “Classification-Based and Energy-Efficient Dynamic Task Scheduling Scheme for Virtualized Cloud Data Center,” *IEEE Trans. Cloud Comput.*, 2021, doi: 10.1109/TCC.2019.2918226.

52. D. Alsadie, “TSMGWO: Optimizing task schedule using multi-objectives grey wolf optimizer for cloud data centers,” *IEEE Access*, vol. 9, pp. 37707–37725, 2021, doi:10.1109/ACCESS.2021.3063723
53. Hussain and M. Aleem, “GoCJ: Google cloud jobs dataset for distributed and cloud computing infrastructures,” *Data*, vol. 3, no. 4, pp. 1–12, 2018, doi: 10.3390/data3040038.
54. M. Chetto and R. El Osta, “Earliest Deadline First Scheduling for Real-Time Computing in Sustainable Sensors,” *Sustainability*, vol. 15, no. 5, pp. 1–18, 2023, doi: 10.3390/su15053972.
55. Z. H. Ebrahim and M. E. Manaa, "Fog-Based Resource Allocation Hybrid ApproachUsing Metaheuristic for Mobile Networks," *2023 6th International Conference on EngineeringTechnology and its Applications (IICETA),* Al-Najaf, Iraq, 2023, pp. 408-413, doi: 10.1109/IICETA57613.2023.10351347
56. G. Nair, G. Hadresh, and V. Pdinesh, “A Comparison Analysis of Fog And Cloud Computing,” *Int. J. Res. Anal. Rev.*, vol. 6, no. 1, pp. 1386–1390, 2019, [Online]. Available: www.ijrar.org