ASJET

Al-Salam Journal for Engineering and Technology
Journal Homepage:
http://journal.alsalam.edu.iq/index.php/ajest
e-ISSN: 2790-4822 p-ISSN: 2958-0862

Optimized Task scheduling in Cloud Environment

Zainab K. Yaser[1]*

[1]Educational Directorate of Thi-Qar faculty of education for pure sciences, University of Thi-Qar, Nasiriyah, Iraq.

Corresponding Author: Zainab K. Yaser

DOI: https://doi.org/10.55145/ajest.2025.04.02.006
Received January 2025; Accepted August 2025; Available online August 2025

ABSTRACT: Cloud computing is a new development in the world of Information Technology (IT) infrastructure and it has brought could of challenges. Task scheduling is one of the main features that allows to be efficient in cloud-computing to guarantee the effectiveness of work with the resources and make the completion time as short as possible. It should, however, be pointed out that the task scheduling in cloud computing belongs to the NP-complete optimization problems. In order to eliminate the difficulties related to task scheduling in cloud computing, a number of algorithms have been presented. One of them is also an original version of the list scheduling scheduling technique, but its implementation is specifically aimed at efficient schedules of tasks execution and load balancing in a cloud environment. This method is based on the Heterogeneous Earliest Finish Time (HEFT) strategy but with some modifications that enhance its efficiency as compared to keeping a similar level of algorithm complexity. I carried out experiments on randomly created Directed Acyclic Graphs (DAGs) with a view to testing the usefulness of the algorithm. The test done on the WorkFlowSim simulator involves testing the real and synthetic workflows. The experiments point out the difference in the effectiveness of the offered mechanism compared to the existing algorithms. As demonstrated by the experiments, the suggested would outperform the current scheduling algorithms by efficiency and use of resources. This algorithm has potential of providing improved results than any prior solutions to the task scheduling problem in computer computing although the task problem is a complex one.

Keywords: Make-span, cloud computing, task scheduling, NP-complete and HEFT

1. INTRODUCTION

Cloud computing is one of the fastest growing branches in the information technology sphere in the recent years. It delivers software, platform, and infrastructure services on-demand, enabling flexible resource utilization. To maintain high service availability and reliability, cloud providers operate multiple data centers distributed across diverse geographical regions. This model allows users to access and deploy applications remotely while benefiting from cost-effective service subscriptions [1]. However, cloud computing infrastructure is not a one-size-fits-all proposition. The infrastructure can be set up in various ways, depending on the cloud provider's approach to building the cloud solution, which is influenced by the specific application. This flexibility is one of the key advantages of using the cloud. If your needs are extensive, operating such servers in-house may be far more costly or challenging than you'd prefer. On the other hand, if you only need a small amount of processing power, purchasing and maintaining a dedicated server might not be desirable. The cloud effectively addresses both of these needs [2]. The cloud computing paradigm is normally divided into three basic service models which include: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) that consists of individual layers of service provision. SaaS is the use of cloud-hosted applications by the clients. For example, a Customer Relationship Management (CRM) application can be used.In PaaS, applications are developed and deployed by customers on cloud infrastructure. The application development tools and programming languages must be supported by the provider. One example is Google Apps.In IaaS, storage, networks, processing power, and other computing resources are provided to customers, and any software—including operating systems and applications—can be installed and run by them [3]. In the IT infrastructures and in the IaaS cloud, cloud users directly utilize essential computing resources such as processing, storage, networks, and others. Virtualization is extensively employed in IaaS to dynamically allocate and break down physical resources, adapting to

*Corresponding author: *zainb.89@utq.edu.iq* the changing resource demands of cloud users [4]. End users can execute their tasks using pay-as-you-go web services from their lightweight mobile terminals. For this, cloud service providers must be able to deploy their clients' services efficiently and automatically. Different technologies, including security, provisioning, and optimization, are developed to address this complicated issue [5].

Cloud computing presents a robust and dependable approach to business computing. This innovative model is designed to provide hosting services and efficiently distribute user requests. cloud computing working over the internet, it has transformed communication, storage, and computing resources into readily available services that operate on a flexible, pay-as-you-go basis [6]. However, it is very difficult to employ such an architecture to address general issues, since the methods used to schedule an application's activities have a significant impact on how efficiently an application may be executed. Because the execution timeframes of traditional scheduling algorithms vary and there may be differences in their communication speeds, making them suboptimal in some cases [7]. One of the important elements to obtain high performance is the effective scheduling of an application's activities on the available resources [8].

Scheduling a (DAG) or task graph on heterogeneous resources is generally classified as an NP-complete problem [9]. To enhance resource utilization, reduce the overall makespan, and ensure balanced workload distribution across cloud infrastructure, task scheduling algorithms allocate user tasks to appropriate cloud resources. These algorithms typically operate under one of two paradigms: static scheduling or dynamic scheduling [10]. Performance, resource management, cost, and other issues are among the difficulties that cloud computing faces, unfortunately. On the other side, task scheduling in cloud computing refers to allocating users' tasks among the available resources in order to increase resource efficacy, decrease execution times, and improve load balancing. Mission scheduling requires the presence of task dependencies.

The problem of scheduling interdependent tasks within heterogeneous computing environments has attracted significant research interest. A prominent focus in this area is the use of (DAGs), which serve as a common representation for modeling the functional dependencies and execution reliability of applications. This form of scheduling, often referred to as DAG scheduling, is specifically designed to handle tasks with dependency constraints [11]. The experiments highlight the effectiveness of the presented mechanism in comparison with the previous algorithms. The experimental results indicate that the proposed algorithm achieves better results than the HEFT, Performance-Efficient Task Scheduling (PPEFT), and QL_HEFT algorithms in terms of makespan and load balancing.

The organization of this paper is as follows: Section 2 provides a comprehensive review of related literature; Section 3 discusses the fundamentals of task scheduling; Section 4 introduces the proposed algorithm in detail; Section 5 presents the experimental results along with a thorough analysis; and Section 6 offers an in-depth discussion of the findings and their implications.

2. LITERATURE REVIEW

This section presents a taxonomy of various task scheduling techniques employed within cloud computing environments [12]. In list scheduling, tasks of the workflow are prioritized, and a higher prioritized task is scheduled prior to another lower one [13]. Topcuoglu [8] presented the two-phase algorithm (HEFT): rank assignment and processor selection. In the first phase, tasks are ranked in descending order based on their priority values. In the second phase, each task is scheduled on the processor that minimizes its earliest finish time.Cui et al. [14] proposed a reinforcement learning-based workflow scheduling algorithm for (DAGs). The approach uses Q-learning to determine task execution time, a reward for the task, and the active hardware and virtual machines.

The algorithm attempts to improve system resource utilization by dynamically responding to cloud condition. Wang et al. [15] proposed a Heterogeneous Task Priority Enhanced Scheduling Algorithm(HSIP) including three primary strategies: (1) a task-priority scheme based on standard deviation considering enhanced weights like computation and communication cost (2) an overlapping entry-task selection policy to reduce the idle time; and (3) a optimization algorithm using Idle Time Slots (ITS) insertion for a better scheduling efficiency.Akbar et al. [16] developed the Median Deviation Dependent Task Scheduling (MDTS) method, which uses the Median Absolute Deviation (MAD) of a task's Estimated Time to Compute (ETC) to rank tasks. The ETC is calculated using a Variation Coefficient (COV) that considers heterogeneity in both tasks and systems.In [17], Dubey and Kumar used a modified form of HEFT to schedule tasks with the same rank and map them onto a heterogeneous processor.

The objective of this approach is to reduce makespan and to attain a superiority over classical HEFT and Critical Path On a Processor (CPOP) algorithms in load balancing, energy consumption and scheduling time.Arif et al. [18] introduced the Priority-Based (PPEFT) in heterogeneous systems. This step consists of a task prioritization stage according to parental rankings in the DAG and a processor assignment stage where tasks are scheduled onto processors according to these priorities.PPEFT improves overall scheduling efficiency and reduces execution cost and time, though load distribution may remain uneven.Gupta [19] introduced the Average Value-based Critical Timing (AVCT) method, which uses average value rankings to select the earliest available time slots. Yu et al [20], suggested a decision tree-based approach for flexible workshop scheduling involving different process plans. Two scheduling strategies based on decision trees were developed for both static and dynamic flexible job shop environments. In the static scenario, all jobs were pre-provided, and a priority dispatching rule was selected using the decision tree to manage each job. In the dynamic environment, jobs were introduced gradually. A rescheduling approach was applied, in which a decision tree was updated regularly to select a priority rule in real time. The objectives considered by this approach included makespan, total flow time, and total latency; however, load balancing across Virtual Machines (VMs) was not addressed.Algorithm—such as Mixed-Criticality Task (MXCT), Minimum Network Configuration Time (MNCT), and Adaptive Voltage and Body-Bias Scaling (AVBS)—were also evaluated. These approaches significantly influence schedule duration and system performance.

3. TASK SCHEDULING

In cloud computing, the term "scheduling" refers to the process of allocating (VMs) to a set of jobs or allocating VMs to execute on the available resources in order to meet customer needs. Utilizing scheduling strategies in a cloud context aims to increase system throughput and load balancing, maximize resource usage, cut expenses, conserve energy, and shorten processing times overall [21]. One of the most vital topics in cloud systems is optimization of job performance. An issue occurs when several users request cloud resources at once; these issues might be resolved by effectively scheduling jobs for available VMs [22]. The scheduling framework allows several apps with dependent tasks to arrive in the cloud system simultaneously. An application queue contains all of the applications. For each program to run quickly before the next one begins, it must be allocated to an appropriate group of virtual machines. An effective scheduler is needed to fit the tasks onto the available resources because there are less resources available than the number of application tasks that have been submitted. Since resource availability is typically limited compared to the number of submitted tasks, an effective scheduler is essential for optimal resource allocation. The information system within the cloud assists the scheduler by providing necessary data for planning task execution and verifying resource availability.Each datacenter is made up of a collection of hosts, and these resources are represented in the datacenter component. Several (VMs) may be made for each host, and the application is run in these VMs [23]. The goal of task scheduling is to effectively assign tasks to the right virtual machines. Tasks can be categorized as independent or dependent depending on their dependencies. The independent tasks don't depend on one another and don't require a priority order to be followed while scheduling. Nonetheless, the dependent tasks must be adhered to during the scheduling process and have a precedence order determined by the dependencies between the activities.Workflow scheduling is the process of scheduling related tasks [24]. Cloud computing, distributed computing, networking, and parallel computing performance are all impacted by resource scheduling and allocation. Several strategies for effectively assigning, scheduling, and scaling cloud resources have been proposed by numerous researchers [25]. Task scheduling is a difficult issue. In order to enhance resource usage in globally dispersed contexts, such as cloud computing, several research looked into job scheduling algorithms and suggested strategies. Finding the best answer requires solving the related NP-hard issue. It has been demonstrated that a variety of heuristic-based approaches offer semi-optimal results. In (IaaS) cloud environments, rule-based heuristic scheduling algorithms are commonly employed due to their faster execution compared to metaheuristic algorithms, which often require extensive computational resources and longer implementation times [26]. The two main categories of job scheduling in a multiprocessor system are deterministic and nondeterministic approaches. Deterministic scheduling, also referred to as compile-time scheduling, is further divided into two subtypes: heuristic-based methods and Guided Random Search-Based (GRSB) techniques. Static scheduling is another name for deterministic task scheduling [19]. Heuristic algorithms depend on the problem and attempt to solve it by applying all aspects of the problem. Their approach is one of discovery and learning, in which a thorough and rigorous search for an ideal answer and a procedure for accelerating reaction time are implemented. However, such algorithms can become trapped in local optima and may fail to find globally optimal solutions [22]. Well-known static task scheduling algorithms include HEFT, Symbiotic Organisms Search (SOS), Fuzzy Genetic (FUGE), and CPOP.

4. PROPOSED ALGORITHM

Here, I present a modified version of the fundamental HEFT method. The study will add some achievements to the already existing body of knowledge in the context of the introduction and the implementation of an improvement in an algorithm to calculate ranks and to choose processors in the rank production stage. During the resource selection phase, I adopt a novel method for generating ranks. Based on computational experiments and evaluations, there is a notable lack of similarity in the work between the original HEFT algorithm experience or work submitted, particularly through the resulting timetable make-span and load balancing. These suggest that the chosen strategy will have a significant impact on the schedule duration. It is also noted that the modified HEFT method uses a maximum computation cost to compute the rankings and choose the least execution time. Our results also show that the suggested approach outperforms the standard HEFT technique in terms of enhanced workflow problem makespan while operating on various virtual computers. The suggested two-phase approach is intended to schedule dependent jobs in a diverse environment. The task prioritization step, which is the initial stage, is essential for task planning. Each job is assigned a priority during this stage based on the ranks established by parental prioritization. A task list is then created by sorting the ranks of each work. The rankings are determined top-down in a (DAG), starting with the original job. The task prioritization phase's planned tasks are used to determine which tasks should be scheduled for the second phase, which is the processor assignment phase. Phase: Calculate the rank. At this level, the ascending rank value should be used to establish the priority of each activity. The following formulas calculate a task's rank improvement recursively. If doing Ti is a last resort for a (sheet) mission, the order of mission Ti is determined using the following rank function:

$$rank_u(T_i) = \max(w_i) + max_{T_{j\in succ(T_i)}} + rank_u(T_J) \quad (1)$$

Where succ (Ti) is the group of tasks' direct successors, Ti (c i,j). The average communication cost of the edge (i, j) is represented, while max(wi) indicates the highest computation cost of mission Ti. As the rank is calculated recursively by moving upward through the mission graph, starting from the exit task, it is known as the ascending order. The phase of selecting a virtual machine, virtual machine selection, the second stage of the proposed technique, involves selecting the best virtual machine for the job, mapping it to that virtual machine, and evaluating its Minimum Execution Time (MET). The presented variables were detailed at the stages of the algorithm and in the following form (algorithm 1):

```
Proposed Scheduling (1) Algorithm
Enter Key: DAG all tasks within one job.
Output: The makespan.
1: Generate a DAG for the mission.
2: All of the mission ti in the diagram (DAG) do…
3: Calculate Rankup(Ti) using equation (1) for each task; the process traverses the DAG upward. Beginning with the final task.
4: End for
5: All tasks are on the list.
6: For minimum execution time, each task is specified for the processor.
7: Calculate the makespan.
8: Get the best scheduling results.
9: End for10: End.
```

It uses Figure.1 as an example of a DAG with communication costs between the three processors' nodes in Table 1.

Table 1 Compution cost mat-rix

Task	VM1	VM2	VM3
T1	9	15	13
T2	11	18	19
T3	10	14	18
T4	12	9	17
T5	11	8	14
T6	12	10	17
T7	10	16	9
T8	8	10	15
T9	17	9	20

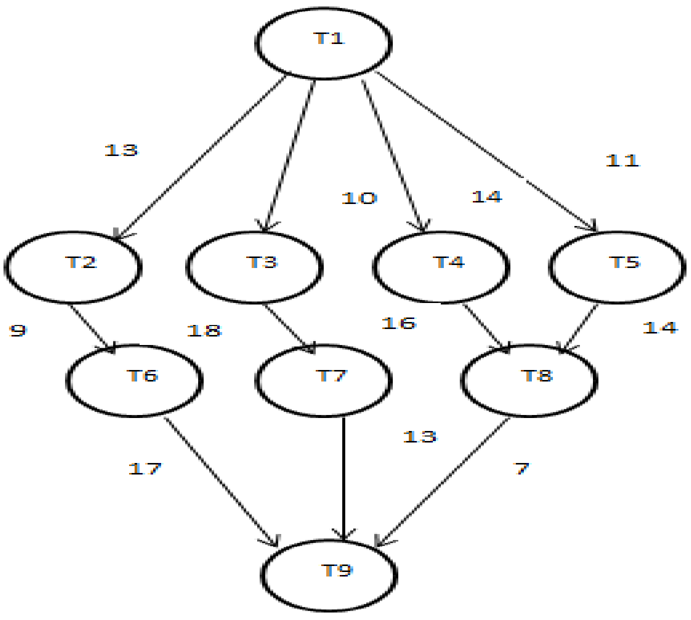

FIGURE 1 **An application example DAG with 9 tasks**

5. EVALUATION OF RESULTS

A comparison analysis of the performance of the suggested algorithm in terms of makespan and load balancing was performed with HEFT, PEFT, QL-HEFT, and other algorithms. I discovered that the suggested method solves the load balancing issue and has a makespan time of 110, which is less than that of the other algorithms (HEFT, PPEFT, and QL-HEFT). Improved load balancing makes sure that no resource is in excess or underutilized. started with task number one (T1) on the list. The job should be assigned to resource P1, as it has the shortest execution time in comparison to P1, P2, and P3.

In the CloudSim simulator, the effectiveness of the suggested algorithm was evaluated, and the measured results show that it reduces the task duration and load balances. Figure 2. Illustrates the implementation outcomes of the suggested HEFT, PEFT, and QL-HEFT. The proposed method surpasses the HEFT, QL-HEFT, and PEFT implementationsThe experimental results demonstrated that the proposed algorithm outperformed existing approaches in terms of makespan efficiency. This was observed across multiple scientific workflows, including Montage, SIPHT, CyberShake, and Epigenomics, evaluated over configurations of 5, 10, 20, and 40 virtual machines, as illustrated in Figure2. The following section highlights comparative outcomes between the original HEFT algorithm and its enhanced variants:

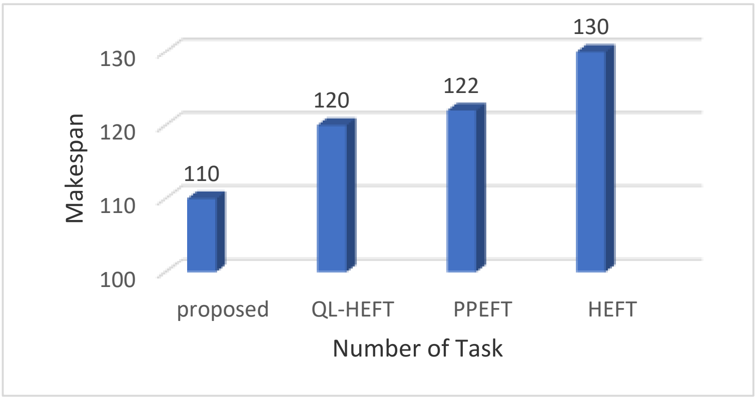

FIGURE 2 **Fulfillment time of the suggested HEFT, PPEFT, and QL_HEFT algorithms**

Figure.3 compares the total makespan of the suggested algorithm to those of the HEFT, PPEFT, and QL_HEFT algorithms. Three processors and nine tasks are used for this comparison. The key variables of the proposed method are convergence speed and efficiency. Figure.4 compares the total schedule length of the proposed algorithm with that of the QL-HEFT, HEFT, and PPEFT algorithms. In this comparison, three processors are used along with 25, 50, and 100 tasks. RTGG was used to produce a random DAG.

This study uses the CloudSim tools to create a simulation environment on a 64-bit Windows i7 computer. The CloudSim toolkit is a discrete event simulator that runs on Java. Cloud computing environments can be modeled and simulated using this well-known methodology. From what has been obtained from experiments and operations, it has been proven that the algorithm gives better results than HEFT, PPEFT, and QL_HEFT algorithms when applied to Montage workflows with 25, 50, 100, and 1000 tasks on processors of different speeds (4, 8, 16, and 32 processors).

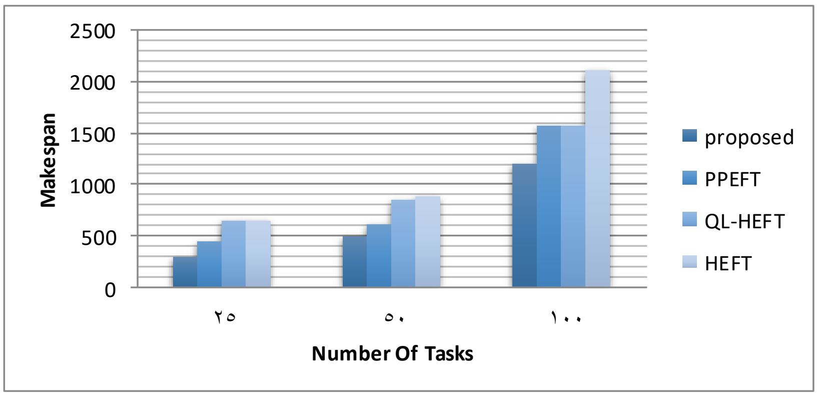

FIGURE 3 Comparison of make-span time HEFT, PEFT , QL-HEFT and proposed algorithm

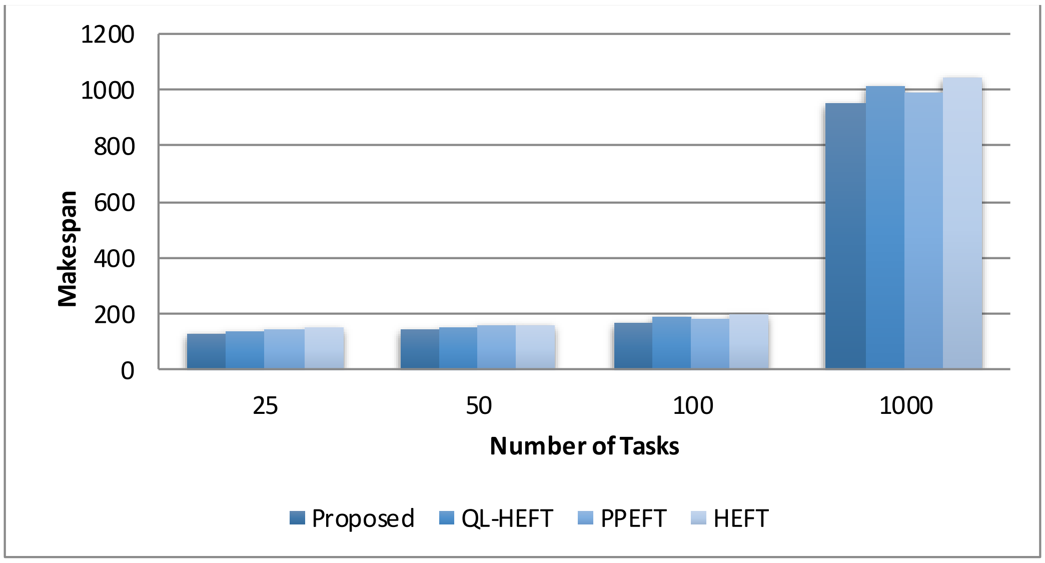

FIGURE 4 Comparative of make-span time HEFT, PEFT, QL-HEFT and suggest algorithm

6. CONCLUSION OF FUTURE WORK

This study introduces an enhanced version of the traditional HEFT algorithm aimed at achieving more optimal solutions for job scheduling challenges within cloud environments. The modifications focus on both the task ranking and resource allocation phases. A distinct methodology is employed to compute task ranks during the prioritization phase, while an improved strategy is proposed for processor selection during the mapping phase. These enhancements to rank calculation and resource assignment represent the primary contributions of the research. The performance of the modified HEFT algorithm was evaluated against the standard HEFT through a series of computational experiments. Findings indicate notable improvements in scheduling efficiency, particularly in terms of load balancing and overall schedule length. The revised method demonstrates superior performance by incorporating the highest task computation cost for ranking and the lowest execution time for processor selection. Overall, the results confirm that the proposed approach significantly reduces makespan compared to the original HEFT, especially when executing complex workflows across multiple VMs in the cloud. I intend to add energy efficiency as a third objective to the algorithm, as there is a trade-off between reducing execution time, balancing load, and reducing energy consumption.

FUNDING

None

ACKNOWLEDGEMENT

The anonymous reviewers are cordially thanked by the authors.

CONFLICTS OF INTEREST

The authors declare no conflict of interest

REFERENCES

[1] S. Mohapatra and B. Majhi, "On solving some issues in cloud computing," unpublished.

[2] A. T. Velte, T. J. Velte, and R. Elsenpeter, Cloud Computing: A Practical Approach. New York, NY, USA: McGraw-Hill, 2010.

[3] E. Salem and K. H. Al-Saedi, "A sample proposal enhancing the security of the cloud computing system through deep learning and data mining," Al-Salam J. Eng. Technol., vol. 3, no. 1, pp. 1–10, Aug. 2023, doi: 10.55145/ajest.2024.03.01.001.

[4] T. Dillon, C. Wu, and E. Chang, "Cloud computing: Issues and challenges," in Proc. 24th IEEE Int. Conf. Adv. Inf. Netw. Appl. (AINA), 2010, pp. 27–33, doi: 10.1109/AINA.2010.187.

[5] K. T. Tran, "Efficient complex service deployment in cloud infrastructure," Ph.D. dissertation, Université de Lyon, 2013.

[6] A. Kaur, P. Singh, R. S. Batth, and C. P. Lim, "Deep-Q learning-based heterogeneous earliest finish time scheduling algorithm for scientific workflows in cloud," Softw. Pract. Exper., vol. 52, no. 3, pp. 689–709, Mar. 2022, doi: 10.1002/spe.2802.

[7] K. R. Shetti, S. A. Fahmy, and T. Bretschneider, "Optimisation of the HEFT algorithm for a CPU-GPU environment," in Proc. 14th Int. Conf. Parallel Distrib. Comput., Appl. Technol. (PDCAT), 2013, pp. 212–218, doi: 10.1109/PDCAT.2013.40.

[8] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," IEEE Trans. Parallel Distrib. Syst., vol. 13, no. 3, pp. 260–274, Mar. 2002.

[9] S. Sandokji and F. Eassa, "Dynamic variant-rank HEFT task scheduling algorithm toward exascale computing," Procedia Comput. Sci., vol. 163, pp. 482–493, 2019, doi: 10.1016/j.procs.2019.12.131.

[10] N. Devi et al., "A systematic literature review for load balancing and task scheduling techniques in cloud computing," Artif. Intell. Rev., vol. 57, art. 276, Sept. 2024, doi: 10.1007/s10462-024-10925-w.

[11] H. Mahmoud, M. Thabet, M. H. Khafagy, and F. A. Omara, "Multiobjective task scheduling in cloud environment using decision tree algorithm," IEEE Access, vol. 10, pp. 36140–36151, 2022, doi: 10.1109/ACCESS.2022.3163273.

[12] S. Yassir, Z. Mostapha, and T. Claude, "E-HEFT: Enhancement of the heterogeneous earliest finish time algorithm for task scheduling based on load balancing in cloud computing," unpublished.

[13] A. Verma and S. Kaushal, "Cost-time efficient scheduling plan for executing workflows in the cloud," J. Grid Comput., vol. 13, no. 4, pp. 495–506, Dec. 2015, doi: 10.1007/s10723-015-9344-9.

[14] D. Cui, W. Ke, Z. Peng, and J. Zuo, "Multiple DAG workflow scheduling algorithm based on reinforcement learning in cloud computing," in Commun. Comput. Inf. Sci., vol. 575, pp. 305–311, 2016, doi: 10.1007/978-981-10-0356-1_31.

[15] G. Wang, Y. Wang, H. Liu, and H. Guo, "HSIP: A novel task scheduling algorithm for heterogeneous computing," Sci. Program., vol. 2016, Art. no. 3676149, 2016, doi: 10.1155/2016/3676149.

[16] M. F. Akbar et al., "List-based task scheduling for cloud computing," in Proc. IEEE Int. Conf. Internet Things (iThings-GreenCom-CPSCom-SmartData), 2016, pp. 652–659, doi: 10.1109/iThings-GreenCom-CPSCom-SmartData.2016.143.

[17] K. Dubey, M. Kumar, and S. C. Sharma, "Modified HEFT algorithm for task scheduling in cloud environment," Procedia Comput. Sci., vol. 125, pp. 725–732, 2018, doi: 10.1016/j.procs.2017.12.093.

[18] M. S. Arif, Z. Iqbal, R. Tariq, F. Aadil, and M. Awais, "Parental prioritisation-based task scheduling in heterogeneous systems," Arab. J. Sci. Eng., vol. 44, no. 4, pp. 3943–3952, 2019, doi: 10.1007/s13369-018-03698-2.

[19] S. Gupta et al., "Efficient prioritization and processor selection schemes for HEFT algorithm: A makespan optimiser for task scheduling in cloud environment," Electronics, vol. 11, no. 16, Art. no. 2557, 2022, doi: 10.3390/electronics11162557.

[20] J. M. Yu et al., "Decision tree-based scheduling for static and dynamic flexible job shops with multiple process plans," J. Korean Soc. Precis. Eng., vol. 32, no. 1, pp. 25–37, 2015, doi: 10.7736/kspe.2015.32.1.25.

[21] S. A. and G. K., "A review on scheduling in cloud computing," Int. J. Ubiquitous Comput., vol. 7, no. 3, pp. 9–15, 2016, doi: 10.5121/iju.2016.7302.

[22] N. Soltani, B. Soleimani, and B. Barekatain, "Heuristic algorithms for task scheduling in cloud computing: A survey," Int. J. Comput. Netw. Inf. Secur., vol. 9, no. 8, pp. 16–22, 2017, doi: 10.5815/ijcnis.2017.08.03.

[23] A. A. Nasr, N. A. El-Bahnasawy, G. Attiya, and A. El-Sayed, "Cost-effective algorithm for workflow scheduling in cloud computing under deadline constraint," Cluster Comput., vol. 26, no. 4, pp. 2961–2978, Dec. 2023, doi: 10.1007/s10586-023-04009-7.

[24] N. Almezeini and A. Hafez, "Review on scheduling in cloud computing," Int. J. Comput. Sci. Netw. Secur., vol. 18, no. 2, pp. 108–111, Feb. 2018.

[25] A. Agarwal and S. Jain, "Efficient optimal algorithm of task scheduling in cloud computing environment," Int. J. Comput. Trends Technol., vol. 9, no. 7, pp. 344–349, Mar. 2014, doi: 10.14445/22312803.

[26] B. A. Al-Maytami, P. Fan, A. Hussain, T. Baker, and P. Liatsis, "A task scheduling algorithm with improved makespan based on prediction of tasks computation time algorithm for cloud computing," IEEE Access, vol. 7, pp. 160916–160926, 2019, doi: 10.1109/ACCESS.2019.2948704.