Using the parallel programming for the workflow scheduling in cloud environment

Peyman Shobeiri
Department of Computer Science
ETS University
peyman.shobeiri.1@ens.etsmtl.ca

Victor Cruz
Department of Computer Science and Software Engineering
Concordia University
victormanuel.cruzhernandez@mail.concordia.ca

I. Introduction

Currently, cloud computing has become the most popular environment for workflow execution by providing customers with on-demand access to computing resources. However, workflow scheduling problems in the cloud environment should consider cloud-native Quality of Service (QoS) such as minimizing monetary costs and reducing execution time.

Due to the complexity of emerging scientific applications, large-scale workflows are used to describe a variety of these applications. A workflow is commonly modeled by a Directed Acyclic Graph (DAG), in which each node represents a computational task, and each edge shows a data dependency between two tasks. The massive amount of computations and data in workflows is required to be processed in a distributed computing environment. Cloud computing is utilized as the latest distributed computing paradigm, providing resources to customers under Service Level Agreement (SLA) rules to ensure the application QoS. Cloud platforms enable users to dynamically lease and release cloud resources for their workflow execution in an on-demand manner and based on the pay-as-you-go payment model.

While cloud computing offers many different layers of abstraction, the workflow model is most compatible with the cloud's infrastructure as service (IaaS) model. In this model, the cloud provides the computing resource to the customer under the SLA to ensure the application's QoS while abstracting the underlying hardware. On the other hand, due to the differences between the traditional distributed systems and cloud environment (e.g., cloud on-demand resource provisioning, faster resources are more expensive, and the pay-as-you-go model), the traditional workflow scheduling algorithms have not been fully fine-tuned for the cloud environment.

Workflow scheduling is assigning resources to workflow tasks in a cloud environment for certain purposes, such as minimizing cost, reducing execution time, or balancing loads across resources. A good algorithm would bring benefits for both cloud providers (e.g., reducing environmental impact) and users (e.g., lowering costs or improving efficiency). Since the problem is NP-hard, we can't find the optimal solution in polynomial time. Therefore, heuristic meta-heuristic or other approaches are necessary.

Multiple research studies have been conducted to address the problem of workflow scheduling in the cloud. Instead of relying on exhaustive search methods, which can be costly in terms of resources, or heuristics that often lead to locally optimal solutions, meta-heuristic algorithms offer a more effective approach. These algorithms can find near-optimal solutions to complex workflow scheduling problems in a reasonable time frame, albeit with higher computational costs.

In this survey, we will explore various algorithms and review papers that utilize different approaches, including parallel programming techniques, to improve workflow scheduling in cloud environments. Implementing parallel programming for workflow scheduling can significantly enhance scheduling speed by enabling tasks to be processed concurrently. This approach is particularly promising for managing the complexity of cloud-based workflows.

II. Workflow Scheduling

Workflow scheduling is a crucial process for managing the execution of workflows in order to achieve user-defined objectives. This involves assigning multiple workflow tasks to suitable computing resources to create an efficient schedule that meets Quality of Service (QoS) requirements. The goal of effective workflow scheduling is to optimize various objectives, including minimizing execution costs, reducing completion times, conserving energy, and maximizing system throughput. User priorities can vary, which makes workflow scheduling a multi-objective optimization problem. For example, some users may prioritize minimizing both execution time and cost while ensuring efficient load balancing across virtual machines (VMs). Others may focus on optimizing energy consumption, reducing delays, or ensuring full utilization of available resources. To address these diverse objectives, robust algorithms are needed to balance trade-offs and meet the specific QoS needs of each workflow application.

III. Different Types of Scheduling

Task scheduling is vital in optimizing resource utilization and performance in computing environments, especially in cloud and edge computing. According to the taxonomy proposed by [21], task scheduling can be broadly divided into two categories: static and dynamic scheduling.

Static scheduling involves the pre-runtime assignment of tasks to resources, often called initial placement. The aim is to devise an optimal deployment plan based on a perfect knowledge of task requirements and resource availability when the scheduling is performed. The approach assumes that the environment remains static and that the resources and workloads will not vary. Since this is a static scheduling done beforehand, speed is not an issue; the emphasis is on achieving an optimal mapping.

On the other hand, dynamic scheduling refers to the dynamic adjustment of task assignments at runtime due to resource availability, bandwidth, or workload condition changes. It gathers real-time information during the execution of tasks to adjust deployments and reschedule them as conditions change, like changes in cloud resources. While dynamic scheduling offers flexibility, it also introduces additional computational overhead in finding and implementing optimal task placements during runtime. That is where parallel approaches are useful since they considerably decrease the time for determining optimum schedules.

IV. Workflow Scheduling in Cloud Computing

With the development of cloud computing, researchers have paid more and more attention to workflow scheduling as an important research area. In this context, workflow scheduling is done to optimize task execution efficiency, considering costs and performance with various constraints.

Among various approaches that have been explored, metaheuristic algorithms and especially their parallel versions are gaining significant attention in handling complex scheduling challenges. The nature of these algorithms inherently makes them suitable for scientific workflows due to their capability of traversing large solution spaces and arriving at near-optimal solutions. In this section, we review several articles and categorize them into three different groups: meta-heuristic, parallel, and hybrid algorithms. We will explain each of these groups in the following subsections.

A. meta-heuristics

Meta-heuristic algorithms are optimization techniques inspired by nature, designed to tackle complex problems by emulating natural processes and behaviors. These algorithms can be categorized into four main types [16]. First, evolution-based methods like Genetic Algorithms (GA) and Memetic Algorithms (MA), etc. Second, physics-based methods such as Simulated Annealing (SA) and Gravitational Search Algorithm (GSA), etc. Third, swarm-based methods like Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO), etc, and finally human-inspired methods such as Tabu Search (TS) and Harmony Search (HS), etc. Metaheuristic algorithms are particularly effective at exploring large solution spaces, making them invaluable for addressing challenging optimization tasks.

PSO has been severely used in many workflow scheduling applications. For example, Pandey et al. [14] proposed a PSO-based approach with the aim of minimizing execution cost. Their method performed better than the Best Resource Selection algorithm and improved the cost reduction by three times. Maria Alejandra Rodriguez et al. proposed a PSO algorithm [17] that is specially designed to minimize the execution cost with deadline constraints in an IaaS cloud. The experimental results of this work indicated that it outperformed other methods when running smaller workflows.

A recent study proposed a multi-objective meta-heuristic approach based on PSO for executing scientific workflows in heterogeneous cloud computing environments [4]. The method focused on two main objectives for workflow scheduling: minimizing makespan and maximizing resource utilization. To improve the PSO algorithm's effectiveness, the researchers introduced an innovative initialization technique and a specialized encoding scheme to efficiently map workflow tasks onto virtual machines (VMs). Experimental results demonstrated that the PSO approach outperformed GA across all tested workflow applications, achieving 100% performance in the evaluated scenarios. However, there were notable limitations in the study. The algorithm did not address the dynamic nature of workflow applications in cloud computing, which is essential for real-world adaptability. Additionally, the evaluation was limited to small and medium-sized workflow tasks, overlooking the impact of larger, more complex workflows. Since small and medium tasks impose limited computational pressure on the scheduling mechanism, the algorithm's true potential and scalability remain uncertain without testing on large-scale workflows. Such tests would provide a more comprehensive assessment of its performance under demanding conditions.

Besides, Genetic Algorithms have also been significantly used in workflow scheduling. Kumar and Verma [7] integrated the Min–Min and Max-Min heuristics with a standard GA for better performance of scheduling, which was able to reduce execution costs and completion times in cloud environments. Furthering this idea, Cheng [12] proposed an enhanced GA approach in terms of cost optimization for hierarchical cloud service workflows, demonstrating the adaptability of GA in optimizing complex scheduling scenarios.

Ant Colony Optimization represents another approach to workflow scheduling, emphasizing the improvements concerning cost and time. In [11] ACO was applied for optimizing makespan; it was shown that execution time was improved by an average of 18.84% compared to the Round Robin algorithm. Quanwang Wu et al. [23] proposed an ACO-based method for task scheduling under deadline constraints. They used the ProLis algorithm to reorder the tasks, which led to a great reduction in execution costs, hence making ACO a very effective approach toward cloud-based scheduling challenges.

A multi-objective scheduling approach known as the Load Balancing Ant Colony Optimization algorithm (LB-ACO) was introduced in [24] to optimize both makespan and workload distribution across cloud resources. This method operates in two distinct phases: the first phase employs Ant Colony Optimization (ACO) to explore the search space and identify optimal solutions, while the second phase uses non-domination sorting to derive a Pareto set of solutions that illustrate the trade-offs between makespan and load balancing. Experimental results have shown that LB-ACO significantly improves system performance, providing a balanced method for workload distribution and execution efficiency. However, a major limitation of ACO-based methods is their dependency on initial conditions, which can greatly affect their ability to find optimal solutions in the search space and influence convergence speed. This reliance underscores the need for further refinement to enhance robustness and ensure consistent performance across different initial configurations.

B. hybrid

In hybrid models, authors mainly use a combination of two metaheuristics or a heuristic combined with a metaheuristic algorithm. For example in [10] a hybrid Genetic Algorithm-Particle Swarm Optimization (GA-PSO) algorithm was introduced to optimize workflow scheduling by minimizing makespan, reducing execution costs, and balancing workloads across cloud resources. The algorithm operates in two phases. In the first phase, the Genetic Algorithm (GA) generates a random population known as chromosomes, which are used as inputs for subsequent optimization. In the second phase, the Particle Swarm Optimization (PSO) method refines the scheduling by iterating over the solutions generated by the GA. This hybrid approach effectively reduces total execution time and cost while distributing workloads evenly across available cloud resources. However, the algorithm has a significant limitation; it only supports a single data center, making it less suitable for large-scale scheduling scenarios that require coordination across multiple data centers. This limitation underscores the need for future enhancements to improve its scalability and applicability in complex cloud environments.

In [27], researchers introduced the Hybrid Harmony Search (HHS) algorithm to tackle the flexible job-shop scheduling problem (FJSP), focusing on minimizing the makespan. The algorithm features a novel conversion method that transforms two-vector codes into continuous vectors, allowing for an effective representation of harmonies during the search process. To enhance its performance further, a local search method was incorporated, enabling the algorithm to refine solutions and improve search efficiency. Simulation results demonstrated that HHS excels in both global and local search capabilities, making it a powerful tool for identifying optimal solutions and significantly reducing makespan. This combination of harmonious representation and local search optimization underscores HHS's effectiveness in addressing complex scheduling challenges.

The Adaptive Scheduling with QoS-Satisfaction (AsQ) algorithm was proposed in [22] to optimize task scheduling in hybrid cloud environments. This algorithm aims to balance the renting costs of public clouds, the utilization rates of private clouds, and the execution times of tasks while adhering to user-defined Quality of Service (QoS) levels. It uses execution time estimations to determine whether a task should be assigned to a public or private cloud. For each job, two schedules are created: one for public cloud resources and another for private cloud resources. Tasks are dynamically allocated to the public cloud if the estimated completion time on the private cloud indicates that the user-defined deadline cannot be met, and vice versa. Experimental results demonstrated that the AsQ algorithm outperformed traditional scheduling approaches such as FIFO, Fair Scheduling, and COSHIC regarding waiting time, execution time, finishing time, and overall QoS satisfaction rate. However, a significant limitation of this method is its failure to account for data transfer costs between public and private cloud resources. This oversight is critical since data transfer latency and expenses can notably affect both execution time and overall cost, potentially impacting the algorithm's practicality and efficiency in real-world applications.

The PCP-ACO algorithm was presented in [18] which is a deadline-constrained workflow scheduling approach designed specifically for cloud environments. Its main goal is to minimize total execution costs while ensuring that user-defined deadlines are met. The PCP-ACO is a hybrid algorithm and operates in two phases based on a list scheduling strategy. In the first phase, workflow tasks are prioritized according to their criticality. In the second phase, each task is allocated to an appropriate resource according to its assigned priority. To achieve its objectives, the algorithm incorporates several innovative steps. It employs a partial critical path method to distribute the overall workflow deadline among individual tasks and constructs a topological sort of workflow tasks to effectively assign priorities. Additionally, it utilizes Ant Colony Optimization (ACO) to allocate resources, ensuring that each task meets its sub-deadline.

C. Parallel Approaches for Task Scheduling

Parallel algorithms have become essential in workflow scheduling, particularly within cloud and distributed computing environments. These algorithms address various challenges related to load balancing, resource utilization, energy efficiency, and cost-effectiveness by using parallelism to enhance scalability and performance. Below, we discuss key contributions that demonstrate the impact of parallel approaches in this field.

Efficient load balancing is a primary focus of parallel algorithms. Anjum and Parveen [1] proposed an optimized load-balancing mechanism for workflows in cloud environments, highlighting the advantages of parallel computing. Their approach improved performance by evenly distributing workloads across available resources, which is crucial for maintaining system stability during high workload periods. Similarly, Fuerst and Sharma [3] introduced a locality-aware load-balancing strategy for serverless clusters, utilizing parallelism to enhance data locality. This minimized communication overhead and improved overall system efficiency.

Cost and energy efficiency are also other critical factors in parallel workflow scheduling. Bisht and Subrahmanyam [2] developed a Load and Cost-Aware Min-Min Workflow Scheduling Algorithm designed for heterogeneous resources in fog, cloud, and edge scenarios. This parallel approach successfully balanced resource usage while reducing execution costs, demonstrating how combining cost awareness with parallelism can be effective. Another study by Malik et al.

[9] presented an energy-efficient load-balancing algorithm that employed queuing and threshold-based techniques for cloud data centers. Their findings illustrated how parallel strategies could significantly decrease energy consumption while maintaining performance.

Kaur and Kalra [6] proposed a hybrid genetic algorithm to optimize scheduling for scientific workflows under deadline constraints, achieving both cost and time savings through parallel execution. In addition, Yang et al. [25] introduced a fully hybrid algorithm for deadline-constrained workflow scheduling in clouds, leveraging parallelism to effectively tackle large-scale, time-sensitive scheduling problems. Toussi et al. [20] presented the EDQWS algorithm, which combined divide-and-conquer techniques with parallel processing to enhance workflow scheduling performance in cloud environments.

Another example of parallel computing in task scheduling is the work of Madej et al. [8], who proposed the utilization of parallel algorithms to optimize dynamic task placement from cloud servers to edge nodes. Qin et al. [15] proposed a reliability-aware multi-objective memetic algorithm specifically designed for scheduling workflows in multi-cloud systems. By leveraging parallel processing, this algorithm effectively balances execution time, cost, and reliability, showcasing the potential of parallelism in addressing complex multi-objective scheduling issues.

A notable contribution from Zambuk et al. [28] involved the development of a hybrid ant colony optimization algorithm aimed at energy-efficient workflow scheduling in cloud computing. This approach combined parallel processing with a multi-objective framework to minimize energy consumption while ensuring that task deadlines were met. The hybrid nature of the algorithm, along with its parallel implementation, proved effective in navigating the energy-performance trade-off in large-scale cloud environments.

Yu et al. [26] introduced a genetic programming-based hyper-heuristic approach for dynamic workflow scheduling in cloud environments. This method used parallelism to adaptively explore scheduling strategies that optimize both cost and performance under dynamic workload conditions. By combining real-time data and using parallel computation, this approach outperformed traditional methods in managing highly dynamic and complex scheduling scenarios (in larger workflows with higher nodes), further highlighting the advantages of parallel approaches in modern cloud environments.

The study referenced in [13] introduced a novel scheduling method called Feasibility Assured TSP-likened Scheduling (FATS), which is designed to improve efficiency, scalability, and applicability in cloud computing environments. FATS is based on the principles of Ant Colony Optimization (ACO) and utilizes a Traveling Salesman Problem-like (TSP-likened) approach to optimize task scheduling. To achieve its goals, the researchers incorporated several innovative techniques. They first developed a generic parallel reconfiguration model that can adapt to various configuration styles, enabling the evaluation of FATS across different systems. Second, they simplified the problem space to enhance accessibility and feasibility, ensuring that solutions could be found more efficiently. Third, they introduced the concepts of T-String and C-String to represent task configurations and scheduling orders, which helped organize and execute tasks effectively. Simulation results showed that FATS outperforms existing methods in robustness, convergence rate, and solution quality, consistently delivering near-optimal results. However, the study only tested the algorithm on small and medium-sized task sets, ranging from 20 to 100 tasks. While these results are promising, the researchers acknowledged that the true efficiency and scalability of FATS need to be validated on larger task sets, such as those involving 1,000 tasks, to better evaluate its capability to handle the demands of high-pressure cloud computing scenarios.

Skinderowicz [19] introduced a GPU-based parallel implementation of the MAX–MIN Ant System (MMAS) to improve the performance of this metaheuristic algorithm. By using GPUs for parallelizing pheromone updates and solution construction, the study significantly reduced execution time while maintaining high-quality solutions. This approach showcased the scalability and efficiency of GPU acceleration in tackling complex optimization problems, making it a useful tool for tasks such as scheduling.

Juarez et al. [5] proposed a dynamic energy-aware scheduling approach for parallel task-based applications in cloud computing environments. This method aims to minimize energy consumption while ensuring the efficient execution of tasks. By dynamically adjusting resource allocation and task scheduling based on real-time energy metrics, the proposed approach optimizes energy efficiency without sacrificing performance. The algorithm adapts to varying workloads and resource conditions, making it well-suited for modern cloud systems, where energy consumption is a critical concern. Through extensive evaluations, the study demonstrated that this dynamic scheduling strategy significantly reduces energy usage while maintaining high levels of task throughput. This offers a balanced solution for energy-aware cloud computing. The work highlights the importance of integrating energy optimization into task scheduling to meet the growing demand for sustainable and efficient cloud operations.

V. Conclusion

In this survey, we explored several research articles that look into the challenge of task scheduling in cloud computing environments. Our study examined various approaches, including metaheuristic algorithms, hybrid techniques, and parallel computing strategies, each offering unique contributions to the optimization of workflow execution. We placed particular emphasis on recent advancements in parallel algorithms, highlighting their potential to improve scalability, efficiency, and adaptability in dynamic, large-scale cloud systems. Throughout the survey, we referenced several significant works to provide a comprehensive overview of the state of the art in task scheduling. We focused on newly proposed methodologies that utilize parallelism to meet the demands of modern cloud computing.

References

[1] Asma Anjum and Asma Parveen. Optimized load balancing mechanism in parallel computing for workflow in cloud computing environment. *International Journal of Reconfigurable and Embedded Systems (IJRES)*, 2023.

[2] Jyoti Bisht and Venkata Vampugani Subrahmanyam. Load and cost-aware min-min workflow scheduling algorithm for heterogeneous resources in fog, cloud, and edge scenarios. *Int. J. Cloud Appl. Comput.*, 12:1–20, 2022.

[3] Alexander Fuerst and Prateek Sharma. Locality-aware load-balancing for serverless clusters. *Proceedings of the 31st International Symposium on High-Performance Parallel and Distributed Computing*, 2022.

[4] Rishabh Gupta, Vatsal Gajera, Prasanta K Jana, et al. An effective multi-objective workflow scheduling in cloud computing: a pso based approach. In *2016 Ninth International Conference on Contemporary Computing (IC3)*, pages 1–6. IEEE, 2016.

[5] Fredy Juarez, Jorge Ejarque, and Rosa M Badia. Dynamic energy-aware scheduling for parallel task-based application in cloud computing. *Future Generation Computer Systems*, 78:257–271, 2018.

[6] Gurpreet Kaur and Maninder Kalra. Cost effective hybrid genetic algorithm for scheduling scientific workflows in cloud under deadline constraint. *International Journal of Advanced Intelligence Paradigms*, 24(3–4):380–402, 2023.

[7] Pardeep Kumar and Amandeep Verma. Scheduling using improved genetic algorithm in cloud computing for independent tasks. In *Proceedings of the international conference on advances in computing, communications and informatics*, pages 137–142, 2012.

[8] Arkadiusz Madej, Nan Wang, Nikolaos Athanasopoulos, Rajiv Ranjan, and Blesson Varghese. Priority-based fair scheduling in edge computing. In *2020 IEEE 4th International Conference on Fog and Edge Computing (ICFEC)*, pages 39–48. IEEE, 2020.

[9] Nimra Malik, Muhammad Sardaraz, Muhammad Tahir, Babar Shah, Gohar Ali, and Fernando Moreira. Energy-efficient load balancing algorithm for workflow scheduling in cloud data centers using queuing and thresholds. *Applied Sciences*, 2021.

[10] Ahmad M Manasrah and Hanan Ba Ali. Workflow scheduling using hybrid ga-pso algorithm in cloud computing. *Wireless Communications and Mobile Computing*, 2018(1):1934784, 2018.

[11] Nasir Mehmood, Muhammad Umer, Riaz Ahmad, and Amer Farhan Rafique. Optimization of makespan and mean flow time for job shop scheduling problem ft06 using aco. *Life Science Journal*, 10(4):477–484, 2013.

[12] Morteza Mollajafari and Hadi Shahriar Shahhoseini. Cost-optimized ga-based heuristic for scheduling time-constrained workflow applications in infrastructure clouds using an innovative feasibility-assured decoding mechanism. *J. Inf. Sci. Eng.*, 32(6):1541–1560, 2016.

[13] Morteza Mollajafari and Hadi Shahriar Shahhoseini. An efficient aco-based algorithm for scheduling tasks onto dynamically reconfigurable hardware using tsp-likened construction graph. *Applied Intelligence*, 45:695–712, 2016.

[14] Suraj Pandey, Linlin Wu, Siddeswara Mayura Guru, and Rajkumar Buyya. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In *2010 24th IEEE international conference on advanced information networking and applications*, pages 400–407. IEEE, 2010.

[15] Shuo Qin, Dechang Pi, Zhongshi Shao, Yue Xu, and Yang Chen. Reliability-aware multi-objective memetic algorithm for workflow scheduling problem in multi-cloud system. *IEEE Transactions on Parallel and Distributed Systems*, 34(4):1343–1361, 2023.

[16] Rajavel Rajakumar, Ponnurangam Dhavachelvan, and Thirumal Vengattaraman. A survey on nature inspired meta-heuristic algorithms with its domain specifications. In *2016 international conference on communication and electronics systems (ICCES)*, pages 1–6. IEEE, 2016.

[17] Maria Alejandra Rodriguez and Rajkumar Buyya. Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds. *IEEE transactions on cloud computing*, 2(2):222–235, 2014.

[18] Peyman Shobeiri, Mehdi Akbarian Rastaghi, Saeid Abrishami, and Behnam Shobiri. Pcp–aco: a hybrid deadline-constrained workflow scheduling algorithm for cloud environment. *The Journal of Supercomputing*, 80(6):7750–7780, 2024.

[19] Rafał Skinderowicz. Implementing a gpu-based parallel max–min ant system. *Future Generation Computer Systems*, 106:277–295, 2020.

[20] Ghazaleh Khojasteh Toussi, Mahmoud Naghibzadeh, Saeid Abrishami, Hoda Taheri, and Hamid Abrishami. Edqws: an enhanced divide and conquer algorithm for workflow scheduling in cloud. *Journal of Cloud Computing*, 11(13):1–33, 2022.

[21] Prateeksha Varshney and Yogesh Simmhan. Characterizing application scheduling on edge, fog, and cloud computing resources. *Software: Practice and Experience*, 50(5):558–595, 2020.

[22] Wei-Jen Wang, Yue-Shan Chang, Win-Tsung Lo, and Yi-Kang Lee. Adaptive scheduling for parallel tasks with qos satisfaction for hybrid cloud environments. *The Journal of Supercomputing*, 66:783–811, 2013.

[23] Quanwang Wu, Fuyuki Ishikawa, Qingsheng Zhu, Yunni Xia, and Junhao Wen. Deadline-constrained cost optimization approaches for workflow scheduling in clouds. *IEEE Transactions on Parallel and Distributed Systems*, 28(12):3401–3412, 2017.

[24] Yang Xianfeng and Li HongTao. Load balancing of virtual machines in cloud computing environment using improved ant colony algorithm. *International Journal of Grid and Distributed Computing*, 8(6):19–30, 2015.

[25] Liang Yang, Yu Xia, Ling Ye, Rui Gao, and Yu Zhan. A fully hybrid algorithm for deadline constrained workflow scheduling in clouds. *IEEE Transactions on Cloud Computing*, 2023.

[26] Yongbo Yu, Tao Shi, Hui Ma, and Gang Chen. A genetic programming-based hyper-heuristic approach for multi-objective dynamic workflow scheduling in cloud environment. *2022 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2022.

[27] Yuan Yuan, Hua Xu, and Jiadong Yang. A hybrid harmony search algorithm for the flexible job shop scheduling problem. *Applied soft computing*, 13(7):3259–3272, 2013.

[28] Fatima Umar Zambuk, Abdulsalam Ya'u Gital, Mohammed Jiya, Nahuru Ado Sabon Gari, Badamasi Ja'afaru, and Aliyu Muhammad. Efficient task scheduling in cloud computing using multi-objective hybrid ant colony optimization algorithm for energy efficiency. *International Journal of Advanced Computer Science and Applications*, 2021.