Metaheuristic Optimization for Dynamic Task Scheduling in Cloud Computing Environments

Longyang Du*, Qingxuan Wang
School of Artificial Intelligence, Jiaozuo University, Jiaozuo 454000, Henan, China

***Abstract*—Cloud computing enables the sharing of resources across the Internet in a highly adaptable and quantifiable way. This technology allows users to access customizable distributed resources and offers various services for resource allocation, scientific operations, and service computing via virtualization. Effectively allocating tasks to available resources is essential to providing reliable consumer performance. Task scheduling in cloud computing models presents substantial challenges as it necessitates an efficient scheduler to map multiple tasks from numerous sources and dynamically distribute resources to users based on their requirements. This study presents a metaheuristic optimization methodology that integrates load balancing by dynamically distributing tasks across available resources based on current load conditions. This ensures an even distribution of workloads, preventing resource bottlenecks and enhancing overall system performance. The suggested method is suitable for both constant and variable activities. Our technique was compared with established metaheuristic methods, including HDD-PLB, HG-GSA, and CAAH. The proposed method demonstrated superior performance due to its adaptive load balancing mechanism and efficient resource utilization, reducing task completion times and improving overall system throughput.**

Keywords*—Dynamic task scheduling; cloud computing; metaheuristic optimization; load balancing; task allocation; resource utilization

I. Introduction

A. Context

Cloud computing is a rapidly evolving technology, marking its place as the next generation in IT and business landscapes [1]. It offers a spectrum of services, including reliable software and hardware, accessible through the Internet and remote data centres [2]. With its architecture, cloud services efficiently manage diverse computing tasks on a large scale, covering multiple IT functions such as storage, computation, database, and application services [3]. The increasing demand for storage, processing, and analysis of extensive datasets has propelled organizations and individuals to embrace cloud computing [4]. Scientific applications, notably those requiring significant computational resources for extensive experiments, have found refuge in cloud deployments due to limitations in local server facilities [5]. Reduced capital costs, immense data generation, and consumption growth from these experiments have driven this shift. Moreover, cloud service providers are now incorporating data parallelism capabilities into their offerings, empowering users to leverage cloud resources and execute their workflows more effectively [6].

B. Problem Statement

Cloud computing is a paradigm that enables universal, flexible, and immediate access to various configurable computing resources in the form of services, applications, storage, servers, and networks, easily delivered and released without much service provider interaction or management effort [7]. It serves as a solution with several advantages to overcome economic and technological challenges. The cloud computing model offers lower total costs and allows companies to concentrate on their primary tasks and functions without concerning themselves with infrastructure issues or the availability and flexibility of resources [8].

Furthermore, the amalgamation of cloud services, including computation, infrastructure, and storage, into the utility model of cloud computing presents an exceptionally appealing environment for scientists to conduct their experiments [9]. Cloud computing provides various service models tailored to meet distinct customer requirements. Cloud service models can be classified as Platform as a Service (PaaS), Software as a Service (SaaS), or Infrastructure as a Service (IaaS) [10]. IaaS offers virtual computing resources over the Internet. It allows users to manage and operate applications without needing to handle physical hardware complexities by combining virtual machines, storage, and networks [11]. PaaS allows customers to develop, run, and manage applications independently of the underlying infrastructure [12]. It includes development frameworks, databases, and tools. SaaS provides subscription-based access to software applications over the Internet [13].

C. Motivation

In recent years, the issue of task scheduling within a distributed environment has become a focal point for researchers. Task scheduling is regarded as a critical concern in the cloud computing domain, taking into account various factors such as completion time, overall cost of executing users' tasks, resource utilization, power consumption, and fault tolerance [14]. The challenge arises in attaining the optimal equilibrium between the time required to complete a task and the amount of energy consumed for a parallel application bound by precedence, resulting in a problem of bi-objective optimization. The resolution to this problem yields a collection of Pareto points, where Pareto solutions indicate that enhancing one target requires making concessions in at least one other objective. Therefore, the resolution to a bi-objective issue comprises a collection of Pareto points rather than a single answer.

Task scheduling in cloud computing environments is commonly known as an NP-complete and multi-objective optimization issue [15]. It deals with the allocation of user-defined tasks on the existing cloud virtual machines. The main goal of any task scheduling strategy is to minimize total execution time. An effective solution to this challenge may be achieved by integrating multiple approaches to enhance task execution and optimize the utilization of resources. This can be achieved by optimizing task placement, task scheduling, and task execution. Additionally, task scheduling algorithms should be adaptive and capable of continuously optimizing their operations in response to changing workloads and resource availability.

D. Contribution

The current investigation centers on implementing and comparing metaheuristic optimization techniques for task scheduling. We compare such methods with conventional heuristics, addressing the problem of scheduling static tasks independently in cloud infrastructure contexts. Experiments are conducted in both uniform and uneven environments. In the uniform scenario, virtual machine characteristics remain constant, whereas the asymmetric environment involves a random selection of virtual machines based on diverse features like MIPS, Bandwidth, and RAM. Despite the simplicity of the symmetric scheduling approach, it fails to fully exploit the potential offered by the asymmetric characteristics of virtual machines. Section I and Section II provide an overview of various conventional metaheuristic task scheduling approaches along with their inherent limitations. Section III gives a comprehensive description of the proposed optimization strategy. Section IV describes the simulation setup and outlines diverse experiments conducted, all grounded in the proposed technique. Finally, Section V articulates the paper's conclusions and suggests potential avenues for future research enhancements applicable to the proposed optimization technique.

II. Related Work

This section discusses existing research efforts addressing task scheduling challenges in cloud computing contexts. Several methodologies have been explored for optimizing the allocation of tasks to virtual machines, enhancing system efficiency, reducing execution times, and maximizing resource utilization. Table I compares various cloud computing task scheduling approaches.

Yang, et al. [16] proposed a task scheduling algorithm derived from game theory in their research. This paper presents three significant contributions tailored to the features of cloud computing. Primarily, leveraging game theory enhances the coordination between task distribution and energy allocation. Secondly, the paper offers a task-scheduling framework to handle big data through a mathematical formulation. Verification by experiment in this research attests to both stable states and optimal computational efficiency.

Chaudhary and Kumar [17] proposed a novel load scheduling technique named Hybrid Genetic-Gravitational Search Algorithm (HG-GSA) with the aim of reducing the overall computational burden, encompassing both execution and transfer costs. HG-GSA employs a hybrid crossover mechanism to explore the optimal arrangement of particles in the search space. The calculated force is then utilized to determine an optimal particle position. The performance of HG-GSA is evaluated against alternative methods using the CloudSim simulator. Through convergence analysis and quantitative assessments, the proposed HG-GSA approach significantly reduces the total computation cost over existing algorithms such as PSO, Cloudy-GSA, and LIGSA-C.

Imene, et al. [18] applied the Non-dominated Sorting Genetic Algorithm (NSGA-III), a third-generation multi-objective optimization strategy, for scheduling cloud computing tasks. They introduced an innovative multi-objective adaptation process designed to optimize three crucial factors: cost, power consumption, and runtime. Further, the study conducted a comparative analysis between NSGA-III and its precursor, NSGA-II, revealing that NSGA-III outperformed NSGA-II.

TABLE I. An Overview of the Recent Cloud Task Scheduling Approaches

References	Algorithm	Contributions	Evaluation metrics
[16]	Game theory-based task scheduling	Mathematical model for big data task scheduling and experimental verification	Equilibrium states and computational efficiency
[17]	Hybrid genetic-gravitational search algorithm	Novel hybrid crossover mechanism	Convergence analysis, statistical assessments, and computation cost reduction
[18]	Non-dominated sorting genetic algorithm	Novel multi-objective adaptation function	Runtime, power consumption, and cost
[19]	Hybrid deadline-constrained, dynamic VM provisioning and load balancing	Hybridization of heuristic techniques with metaheuristic	Makespan, cost, and VM utilization
[20]	Context-aware adaptive heuristic-based mechanism	Context-aware adaptive heuristic-based solution and significant performance improvements	Performance efficiency and energy savings
[21]	Adaptive ant colony optimization algorithm	Pheromone adaptive update mechanism	Task completion time, execution cost, and balance degree
[22]	Moth search algorithm with differential evolution	Strong exploration and exploitation	Makespan
[23]	Chemical reaction partial swarm optimization	Integration of chemical reaction optimization and partial swarm optimization	Execution time, makespan, cost, and energy
[24]	Deep Q-learning network	Utilization of deep Q-learning network	Makespan, SLA violation, and energy consumption

Kaur and Kaur [19] proposed a hybrid delay-constrained dynamic virtual machine provisioning and load balancing approach called HDD-PLB. The primary goal of HDD-PLB is to enhance VM utilization by achieving uniform load distribution. This optimization strategy relies on combining heuristics with metaheuristics to attain optimum performance, focusing on metrics such as cost and makespan. Within the HDD-PLB methodology, two heuristics are proposed: hybrid heterogeneous earliest finish time heuristic with Ant Colony Optimization (ACO) algorithm and hybrid predicted earliest finish time heuristic with ACO algorithm. A comprehensive analysis and comparison of these approaches is conducted to determine their superiority within the proposed HDD-PLB model.

Kulkarni and Annappa [20] proposed an effective context-aware adaptive heuristic-based (CAAH) methodology tailored for virtual machine allocation in diverse and heterogeneous cloud data centers. CAAH accounts for both the inherent properties of physical machines and the varying load conditions (moderate or high) within heterogeneous data centers. The primary objective is to augment performance efficiency and facilitate power savings for operators managing data centers. Through experimental assessments employing both genuine cloud workloads and synthetic workloads, noteworthy enhancements in performance and energy conservation were observed with CAAH in comparison to a widely recognized adaptive heuristic-based technique.

Liu [21] proposed a dynamic task-scheduling technique designed for cloud computing and based on the ACO algorithm. Their proposed approach enhances the standard ACO by integrating pheromone adaptive updating to expedite convergence while effectively circumventing local optima. This enhanced algorithm generates a distribution scheme that offers reduced processing time, minimized costs, and well-balanced task loads based on user-submitted tasks. By conducting experiments on a cloud computing platform, the traditional ACO is compared against the enhanced adaptive ACO algorithm. The empirical data illustrates that the improved adaptive ACO efficiently identifies optimal solutions for cloud computing resource scheduling issues, resulting in reduced task completion times, decreased execution costs, and maintaining a balanced load across the cloud system.

Abd Elaziz, et al. [22] introduced an innovative approach to solving the cloud task scheduling challenge with a primary focus on minimizing the amount of time needed to schedule diverse tasks across distinct virtual machines. The proposed methodology incorporates the Differential Evolution (DE) technique into the Moth Search Algorithm (MSA). The MSA draws inspiration from moth navigation toward a light source, a natural process, leveraging Levy flights and phototaxis to emulate exploitation and exploration capabilities. While the MSA exhibits robust exploration abilities, its exploitation facet requires enhancement, prompting the integration of DE as a local search technique. Three experiments were performed to measure the effectiveness of the newly introduced MSDE algorithm. The initial test compares the performance between the classic MSA and the modified algorithm across twenty global optimization problems. In the subsequent two testing phases, the proposed algorithm was benchmarked with various heuristic and meta-heuristic algorithms, utilizing both synthetic and real-world data.

Dubey and Sharma [23] introduced a pioneering task scheduling approach, termed Chemical Reaction Partial Swarm Optimization (CRPSO), to allocate several independent tasks to available virtual machines. This innovative method combines partial swarm optimization and chemical reaction optimization, amalgamating their features to sequence the optimal task schedule based on demand and deadlines. The aim is to enhance quality across various factors such as cost, energy, and makespan. Their simulation experiments, conducted via the CloudSim toolkit, confirm the performance of the proposed algorithm. Comparative tests, varying the number of tasks and virtual machines, demonstrate an average reduction in execution time ranging between 1% to 6%, exceeding 10% in certain scenarios. The makespan results also exhibit an effectiveness enhancement between 5% to 12% and a total cost reduction between 2% to 10%, while the energy consumption rates show an improvement of 1% to 9%.

Mangalampalli, et al. [24] utilized a multi-dimensional deep learning algorithm to manage the cloud task scheduling issue, conducting extensive simulations through the Cloudsim toolkit. The simulations were executed in two phases: first utilizing randomly generated workloads and then incorporating HPC2N and NASA workloads to assess the efficiency of the suggested algorithm. The proposed scheduler was compared against conventional schedulers like Earliest Deadline First, RR, and FCFS.

III. Proposed Method

The client provides a set of tasks, aiming to generate an optimal task execution plan using a metaheuristic method based on optimization techniques. One vital aspect of any meta-heuristic algorithm in achieving an optimal solution is the selection of a seed arrangement. Arrangements of seeds serve as initial feasible solutions to the problem, aiding optimization algorithms in the quest for an optimal solution. These arrangements play a critical role in the rapid convergence of any optimization-based solution. Researchers have employed various strategies to generate seed arrangements, depending on the nature of the problem. These strategies encompass selecting a complex arrangement, a logical configuration based on a particular problem model, or a heuristic-based arrangement. Each approach has its own advantages and drawbacks. However, in many cases, an uneven seed arrangement is utilized to generate a seed arrangement in the absence of a proper heuristic.

The proposed algorithm aims to minimize execution time and meet deadlines while preserving task dependencies across various users. It operates based on a Directed Acyclic Graph (DAG) representing the task set (T) and task dependencies through edges denoting data transmission time. These relationships establish entry-exit dependencies between child and parent nodes, forming the basis of the task behavior. The proposed algorithm operates as a population-based approach, aligning with swarm intelligence behavior to provide an optimized solution to complex data. It functions as a meta-heuristic technique in comparison to other algorithms. The algorithm encompasses two phases: scheduling jobs with static

constraints and dynamic constraints. These phases aim to handle task positions in the schedule based on execution time and deadlines, with a primary goal of minimizing the makespan. The algorithm considers the constraints stated by the user and adjusts the tasks accordingly, ensuring dependencies are maintained throughout the sorting process.

During the task scheduling process, the algorithm utilizes swarm behavior, mimicking the distribution and interaction patterns of particles for improved optimization of multi-objective tasks. Its adaptability helps in solving a wide range of NP-hard-level tasks, effectively handling the scheduling of various tasks on different machines. The algorithm's efficiency is further enhanced by its ability to detect all scheduled tasks, ensuring effective outcomes. Employing a random phase, the algorithm aims to optimize the scheduling of cloudlets for execution on Virtual Machines (VMs). Particle fitness, bandwidth, MIPS, flow time, response time, resource usage, throughput time, and imbalance degree guide the selection of particles in the pursuit space, ensuring improved wellness values and effective execution outcomes.

The aim is to allocate a set of tasks $(T = T_1, T_2, T_3, \dots, T_n)$ onto a designated group of processors $(P = P_1, P_2, P_3, \dots, P_n)$ within a cluster of VMs. This task allocation, called solution S, follows predetermined measures and constraints within the cloud environment.

$$F(S) = min \sum_{i=1,j=1}^{t,m} Ct \quad (1)$$

In Eq. (1), *F(S)* represents the fitness function of the solutions, *m* corresponds to the total number of available machines, *t* stands for the entire number of tasks submitted by the user, and *Ct* denotes the completion time of all tasks. Fitness function values vary with the type of job. Jobs can be categorized as either dynamic or static. Static jobs possess predefined properties, such as a fixed total data amount, data flow within the system, and time constraints. Conversely, dynamic jobs encompass undefined job properties, like data bursting and indeterminate data types. Users are required to specify whether the job properties are static or dynamic when submitting the data.

For static job scheduling, where job properties are determined by the cloud service provider, denoted as $(P_1, P_2, P_3, \dots, P_n)$. The user-provided variables are used to characterize the present infrastructure utilization and determine the needed virtual machine cluster. Various categories of VM clusters are evaluated, and their corresponding fitness scores are computed. By employing the optimal fit algorithm, the VM cluster that is most appropriate is chosen, thereby accomplishing load balancing. The cost of task execution is determined by the user's given property values, as per Eq. (2). Fig. 1 depicts the system architecture of the suggested approach.

$$C = \sum_{i=1,j=1}^{n,m} Pi \quad (2)$$

In Eq. (2), *Pi* represents the property of the i^{th} job, *n* defines the number of job properties, *m* denotes the number of jobs, and *C* refers to the cost of executing the function. Job costs are considered when computing fitness values for potential VM clusters, and these values are used to pick the most appropriate VM cluster. Eq. (3) details the fitness function calculation.

$$F(S) = min \sum_{i=1,j=1,k=1}^{n,m,p} (P_1, P_2, \dots, P_x) + T_j + M_k \quad (3)$$

Where *F(S)* represents the fitness value of the respective cluster, *p* indicates the available machines within the respective cluster, *M* is the machine, *m* signifies the number of tasks, *T* refers to the corresponding task, *x* reflects the job's property, and *n* indicates the number of jobs. The proposed algorithm defines the static scheduling of tasks, as outlined in the Algorithm. 1.

Algorithm. 1. Pseudocode for the proposed static task scheduling

```
Function ProposedAlgorithm(J, P, M):
    Initialize BestFitCluster as empty
    For Each Task Ti in Job J:
        For Each VM Cluster Mk in Set of VM Clusters:
            Calculate Cost(C) for placing Ti in Mk based on properties
P
        Select VM Cluster M with minimum Cost(C)
        If BestFitCluster is empty or Cost(C) < Cost(BestFitCluster):
            Update BestFitCluster with M
    Return BestFitCluster
End Function
```

Dynamic task scheduling involves the possibility of unspecified task properties, which necessitates the service provider to establish a minimal set of parameters and their assigned weights. The user provides these variables during the first task submission, which determines the allocation of required machines. The quantity of machines in operation can be modified according to the workload duration of the task. The cost of task execution is determined by the highest value that the user is willing to pay for the first setup, as specified in Eq. (4).

$$C = max \sum_{i=1,j=1}^{n,m} Pi \quad (4)$$

Where Pi represents the maximum value of the property, while *n* signifies the number of job properties, and *m* represents the task count. Throughout runtime, the highest property estimates gathered from the job's tasks are logged. The average value of the property is considered when allocating a new VM cluster for arriving tasks carrying varying properties, determined by Eq. (5) and Eq. (6).

$$Pi(T) = {\left(\sum_{i=1}^{n} Vi\right)}/{n} \quad (5)$$

$$F(S) = min\left(\sum_{i=1,j=1,k=1}^{n,m,p} (P_1, P_2, \dots, P_x) + T_j + M_k\right) < max \sum_{i=1,j=1}^{n,m} Pi \quad (6)$$

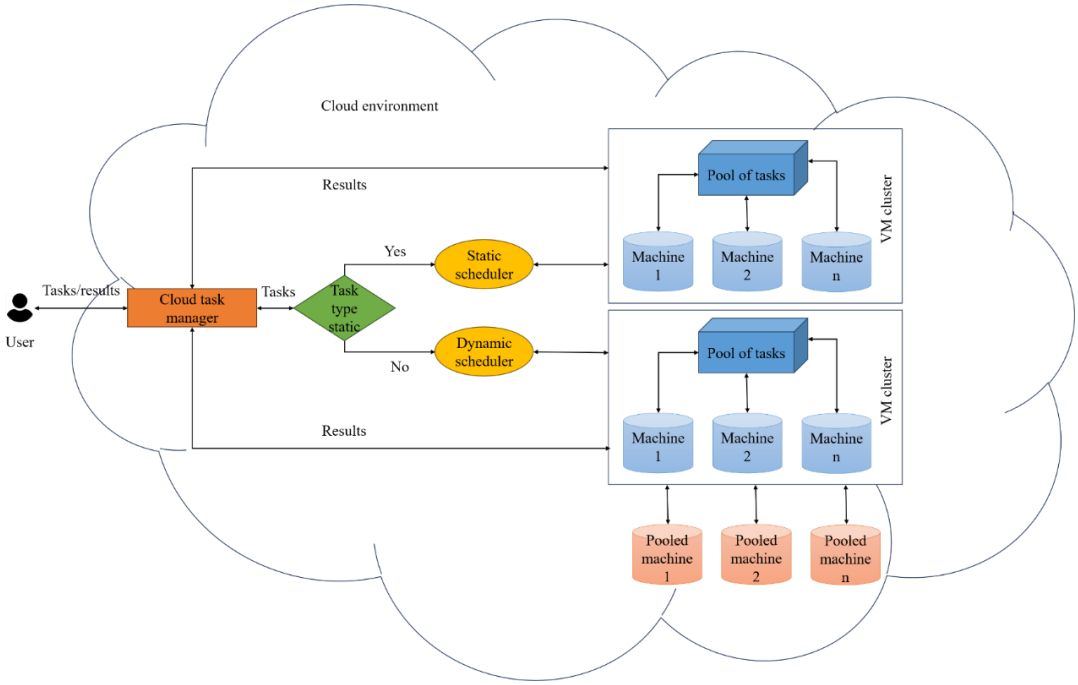

Fig. 1. System architecture.

Eq. (5) defines *Pi(T)* as the average property value of the job, *Vi* as the specific property value, and *n* as the total number of property values obtained from the operations. The fitness value of the VM cluster is denoted by *F(S)*. In this equation, *n* refers to the number of jobs, *x* stands for the value of property *P* for the job, *T* is the task, *m* signifies the number of tasks, *M* represents the machine, and *p* refers to the number of machines within the VM cluster. The computed value should be the minimum among all fitness values lower than the maximum value the user is willing to accept for task execution. The method proposed in Algorithm 2 defines the dynamic scheduling of tasks.

Algorithm 2. Pseudocode for the proposed dynamic task scheduling

Start
Input: Job with set of Tasks J(T1, T2, ….. Tj);
Set of task properties (P1, P2, ……Pn);
Set of VM cluster properties (M1, M2, …… Mk);
Set of Task property values V;
Output: Bestfit VM cluster F(S)
Initialize BestfitCluster to null
For each task T in J
Calculate Cost C for each VM cluster using task properties P and values V
For i = 1 to n & j = 1 to m
Calculate Fitness F(S) for each VM cluster
end For
Find the VM cluster with the minimum Fitness (min F(S))
If BestfitCluster is null or F(S) < Fitness of BestfitCluster
Update BestfitCluster with the current VM cluster
End For
Return BestfitCluster
End

IV. Results and Discussion

The efficiency of the suggested technique has been evaluated using the CloudSim simulator. Table II provides a concise overview of the specifications for key components of a cloud platform, including virtual machines (VMs), cloudlets, data centers, and clients. These parameters are regarded as minimal for dynamic simulation and constant for static simulation, allowing for changes during execution. The suggested method is evaluated against three techniques described in Section II, specifically HDD-PLB [19], HG-GSA [17], and CAAH [20]. The performance metrics considered include VM utilization ratio, execution time, response time, and makespan time. VM utilization ratio denotes the number of VMs deployed relative to the total VMs available, execution time indicates the duration of task completion in the VMs, response time signifies the scheduler's time taken to schedule tasks, and makespan represents the finishing time of the last task.

TABLE II. Simulation Parameter Specifications

Components	Attributes	Values
Task	Total number	250-2000
Cloudlet	Length	50,000
	Total number	500
Host	Bandwidth	10 GB/s
	Storage	100 GB
	RAM	2 GB
	Total number	3
VM	Total number	100
	OS	Linux
	MIPS	10000
	Processor	2.4 GHz
	SSD	100 GB
	RAM	2 GB
Data center	Total number	2

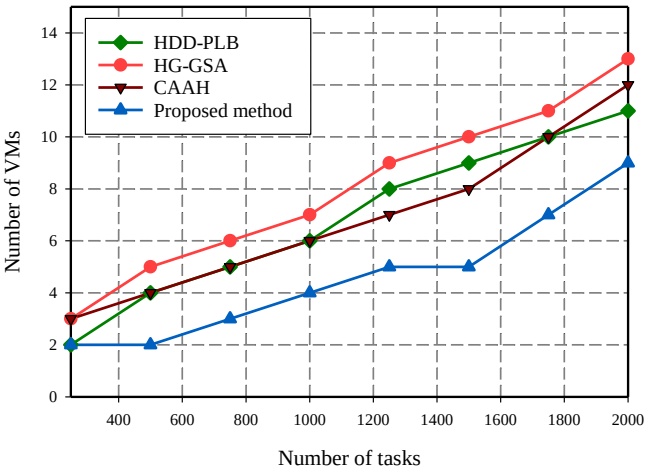

Fig. 2. VM utilization ratio comparison.

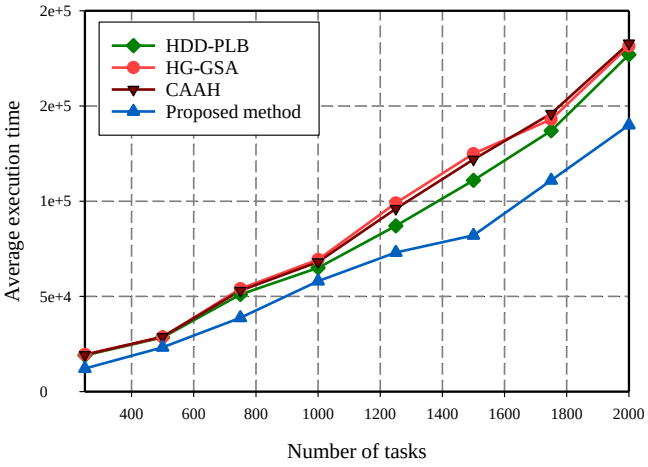

Fig. 3. Execution time comparison.

Fig. 2 compares the suggested strategy with existing alternatives in terms of the VM utilization ratio. This indicator is crucial for evaluating the efficiency and performance of resource allocation in a cloud computing environment. It offers valuable information on the efficient utilization of resources by virtual machines, enabling the discovery of virtual machines that are either underutilized or overutilized while making appropriate modifications to achieve optimal performance. The figure illustrates that the proposed algorithm schedules tasks optimally, using a lower number of VMs compared to the comparative algorithms. Specifically, for tasks based on static properties, our algorithm minimizes the number of VMs required, outperforming other methods. The performance comparison indicates that our algorithm optimizes VM scheduling and achieves load balancing, enabling the cloud service provider to handle more tasks from different users in real-time. However, a drawback of the proposed approach is its tendency to utilize more VM placements when dealing with dynamically changing tasks, making it challenging to predict future VM usage accurately.

Fig. 3 illustrates a comparative analysis of our approach and other algorithms based on their respective execution times. The data presented in the figure proves that our technique, which supports both active and passive property-based tasks, outperforms other present ones through optimized VM placement, resulting in improved execution time. By efficiently allocating fewer complex tasks to VMs, our technique reduces the overall execution time.

In Fig. 4, the proposed algorithm is compared to other algorithms for response time. This comparison shows that our algorithm converges faster and delivers VM placement scheduling for the given tasks more rapidly. It can promptly allocate VMs by prioritizing static property values for simpler tasks, enhancing its ability to effectively handle tasks based on dynamic properties.

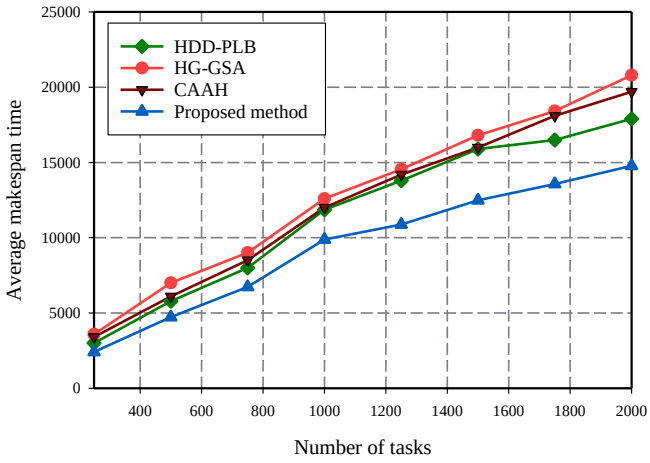

Fig. 4. Response time comparison.

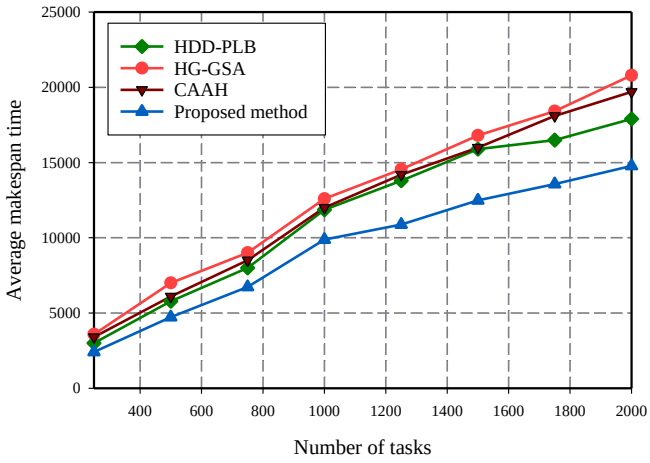

Fig. 5. Makespan time comparison.

Fig. 5 compares the proposed algorithm with existing methods measured by the makespan metric. The figure indicates that our algorithm offers optimal placement of VMs regardless of increasing workloads compared to the comparative algorithms. The algorithm considers task properties to place VMs optimally.

Despite all these improvements in task scheduling and resource allocation, the proposed method has some limitations. One primary limitation is the overhead resulting from more VM placements to accommodate dynamically changing tasks. While the method is superior in terms of efficiency in carrying out tasks characterized by static properties of resources, it can over-allocate VMs in such a task with dynamic properties. This tendency can bring about some issues, including reduced efficiency and increased operation costs since forecasting the future usage of VMs becomes complicated. As such, the method does not guarantee the selection of an inexpensive solution that would be advantageous in systems with volatile traffic loads.

Another limitation is that complexity can result from handling scheduling algorithms employed in the examination process. Although load balancing coupled with metaheuristic optimization techniques offers a solution for resolving inefficiencies, it is accompanied by increased computational complexity. This complexity might affect the method's usability when scaling up and applying it to larger and more diverse cloud environments. Further, due to the dependency of the algorithm on chosen performance parameters and simulation parameters, it is possible to infer that the efficiency of the algorithm can be different from that of another Cloud platform and from that of a real-world scenario. Mitigating these limitations involves an enhancement of the method and its application with the aim of achieving uniformity and efficiency in various settings and projects.

V. Conclusion

Cloud computing has gained popularity due to its flexible and resourceful nature, providing adaptable resources on a shared infrastructure. This technology provides a framework for various services, from scientific operations to service computing, highlighting the critical role of efficient task scheduling in ensuring optimal resource allocation and performance. In this study, we introduced a novel approach for task scheduling in cloud infrastructure services, addressing the significant challenge of mapping tasks to available resources while minimizing execution plan objectives. The proposed technique leverages a metaheuristic optimization method along with load distribution, designed to optimize cloud computing service providers' overall performance and effectively alleviate scheduling issues. One of the key strengths of our proposed approach is its adaptability to both static and dynamic task conditions. In static scenarios, where VM parameters are fixed, and in dynamic conditions, where parameters are adjusted in real-time, our method exhibits efficacy and flexibility. Simulation results unequivocally demonstrated the superiority of our proposed technique over existing methods, showcasing notable improvements in key performance metrics such as makespan, response time, and execution time.

The outcomes of our study underscore the practical viability and potency of our proposed metaheuristic optimization approach with load balancing in addressing the complexities of task scheduling in cloud infrastructure services. It not only optimizes resource utilization but also contributes significantly to enhancing user experience by ensuring guaranteed performance and efficient resource allocation. While this research presents promising results, future work could delve deeper into exploring the scalability of the proposed method for larger and more diverse cloud environments. Moreover, considering real-world deployment and testing on varied cloud platforms could further validate the applicability and robustness of the approach. Overall, the proposed technique stands as a promising step towards addressing the challenges in task scheduling within cloud infrastructure services, offering a potential avenue for further advancements and practical implementations in the field.

References

[1] V. Hayyolalam, B. Pourghebleh, A. A. P. Kazem, and A. Ghaffari, "Exploring the state-of-the-art service composition approaches in cloud manufacturing systems to enhance upcoming techniques," The International Journal of Advanced Manufacturing Technology, vol. 105, no. 1-4, pp. 471-498, 2019.

[2] K. Saidi and D. Bardou, "Task scheduling and VM placement to resource allocation in Cloud computing: challenges and opportunities," Cluster Computing, vol. 26, no. 5, pp. 3069-3087, 2023.

[3] W. Wang and Z. Liu, "Cloud Service Composition using Firefly Optimization Algorithm and Fuzzy Logic," International Journal of Advanced Computer Science and Applications, vol. 14, no. 3, 2023.

[4] S. Zhao, J. Miao, J. Zhao, and N. Naghshbandi, "A comprehensive and systematic review of the banking systems based on pay-as-you-go payment fashion and cloud computing in the pandemic era," Information Systems and e-Business Management, pp. 1-29, 2023.

[5] X. Liu and Y. Deng, "A new QoS-aware service discovery technique in the Internet of Things using whale optimization and genetic algorithms," Journal of Engineering and Applied Science, vol. 71, no. 1, p. 4, 2024.

[6] B. Kruekaew and W. Kimpan, "Multi-objective task scheduling optimization for load balancing in cloud computing environment using hybrid artificial bee colony algorithm with reinforcement learning," IEEE Access, vol. 10, pp. 17803-17818, 2022.

[7] B. Pourghebleh, A. A. Anvigh, A. R. Ramtin, and B. Mohammadi, "The importance of nature-inspired meta-heuristic algorithms for solving virtual machine consolidation problem in cloud environments," Cluster Computing, pp. 1-24, 2021.

[8] A. G. Gad, E. H. Houssein, M. Zhou, P. N. Suganthan, and Y. M. Wazery, "Damping-assisted evolutionary swarm intelligence for industrial iot task scheduling in cloud computing," IEEE Internet of Things Journal, 2023.

[9] M.-L. Chiang, H.-C. Hsieh, Y.-H. Cheng, W.-L. Lin, and B.-H. Zeng, "Improvement of tasks scheduling algorithm based on load balancing candidate method under cloud computing environment," Expert Systems with Applications, vol. 212, p. 118714, 2023.

[10] M. Yadav and A. Mishra, "An enhanced ordinal optimization with lower scheduling overhead based novel approach for task scheduling in cloud computing environment," Journal of Cloud Computing, vol. 12, no. 1, p. 8, 2023.

[11] H. Godhrawala and R. Sridaran, "Apriori Algorithm Based Approach for Improving QoS and SLA Guarantee in IaaS Clouds Using Pattern-Based Service-Oriented Architecture," SN Computer Science, vol. 4, no. 5, p. 700, 2023.

[12] F. Thabit, O. Can, R. U. Z. Wani, M. A. Qasem, S. Thorat, and H. A. Alkhzaimi, "Data security techniques in cloud computing based on machine learning algorithms and cryptographic algorithms: Lightweight algorithms and genetics algorithms," Concurrency and Computation: Practice and Experience, p. e7691, 2023.

[13] P. A. Malla and S. Sheikh, "Analysis of QoS aware energy-efficient resource provisioning techniques in cloud computing," International Journal of Communication Systems, vol. 36, no. 1, p. e5359, 2023.

[14] I. Behera and S. Sobhanayak, "Task scheduling optimization in heterogeneous cloud computing environments: A hybrid GA-GWO approach," Journal of Parallel and Distributed Computing, vol. 183, p. 104766, 2024.

[15] P. V. Reddy and K. G. Reddy, "An energy efficient RL based workflow scheduling in cloud computing," Expert Systems with Applications, vol. 234, p. 121038, 2023.

[16] J. Yang, B. Jiang, Z. Lv, and K.-K. R. Choo, "A task scheduling algorithm considering game theory designed for energy management in cloud computing," Future Generation computer systems, vol. 105, pp. 985-992, 2020.

[17] D. Chaudhary and B. Kumar, "Cost optimized hybrid genetic-gravitational search algorithm for load scheduling in cloud computing," Applied Soft Computing, vol. 83, p. 105627, 2019.

[18] L. Imene, S. Sihem, K. Okba, and B. Mohamed, "A third generation genetic algorithm NSGAIII for task scheduling in cloud computing," Journal of King Saud University-Computer and Information Sciences, vol. 34, no. 9, pp. 7515-7529, 2022.

[19] A. Kaur and B. Kaur, "Load balancing optimization based on hybrid Heuristic-Metaheuristic techniques in cloud environment," Journal of King Saud University-Computer and Information Sciences, vol. 34, no. 3, pp. 813-824, 2022.

[20] A. K. Kulkarni and B. Annappa, "Context aware VM placement optimization technique for heterogeneous IaaS cloud," IEEE access, vol. 7, pp. 89702-89713, 2019.

[21] H. Liu, "Research on cloud computing adaptive task scheduling based on ant colony algorithm," Optik, vol. 258, p. 168677, 2022.

[22] M. Abd Elaziz, S. Xiong, K. Jayasena, and L. Li, "Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution," Knowledge-Based Systems, vol. 169, pp. 39-52, 2019.

[23] K. Dubey and S. C. Sharma, "A novel multi-objective CR-PSO task scheduling algorithm with deadline constraint in cloud computing," Sustainable Computing: Informatics and Systems, vol. 32, p. 100605, 2021.

[24] S. Mangalampalli, G. R. Karri, M. Kumar, O. I. Khalaf, C. A. T. Romero, and G. A. Sahib, "DRLBTSA: Deep reinforcement learning based task-scheduling algorithm in cloud computing," Multimedia Tools and Applications, pp. 1-29, 2023.