Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2025 SCPE. Volume 26, Issues 2, pp. 973–982, DOI 10.12694/scpe.v26i2.3911

EXPLORATION ON RESOURCE SCHEDULING OPTIMIZATION STRATEGIES IN CLOUD COMPUTING ENVIRONMENT

XIAOLAN LI* AND WENFU YANG†

Abstract. The rapid expansion of cloud computing has brought opportunities and difficulties for resource schedule optimization to enhance productivity and decrease expenses. This study offers a thorough analysis of resources planning optimisation techniques in cloud computing settings, emphasizing the most recent developments and approaches. First, we go over the basic ideas of resource scheduling, such as resource sharing, load balancing, and task distribution. The task scheduler's execution-time mappings among the future demand and cloud resources makes application planning one of the major difficulties in cloud computing. By decreasing makespan and improving resource utilization, an effective scheduling system is required to schedule the diversified workload and enhance performance metrics. Numerous scheduling methods that were previously in use only considered makespan and utilization of resources metrics, ignoring other important factors that have an immediate effect on cloud service performance, such as energy consumption and migration time. To address the problems, the authors have developed the Hybridize Whale Optimization Algorithm (H-WOA), a nature-inspired multi-objective task scheduling algorithm that can make scheduling decisions at runtime depending on the availability of cloud resources and impending workload requirements. Furthermore, the suggested method distributes the resources according to task priorities and end users' budgets. The workload for the proposed H-WOA technique, which is based on the Cloudsim toolkit, is created by creating datasets (da01, da02, da03, da04) with various task densities and workload records from NASA (da05, da06) and HPC2N (da01, da02) parallel workload repositories. An extended experiment's findings demonstrate that the suggested H-WOA technique enhanced the important variables and performed better than alternative baseline policies.

Key words: Resource Scheduling, Optimization, Cloud Computing Environment, Hybridize Whale Optimization Algorithm

1. Introduction. The ability to offer resources and services on demand over fast computer networks is one of the main advantages of cloud computing. As network technologies and infrastructure continue to evolve and advance, cloud computing has proven to perform better at handling various large-scale and complicated end-users' (clients') workloads. Modern physical and virtualized technology, coupled with advanced development platforms and applications, satisfy the needs of heterogeneous clients. The cloud and the internet of things (IoT) have emerged as key ideas that establish new standards for industry advances at the current stage of technological breakthroughs [4].

The term "cloud computing" describes the provision of several computer services via the internet, including networking, processing, and storage [8, 13]. Individuals and organizations can now access and use computing resources without having to manage or own the hardware that underlies them because to this approach to change in architecture. Rather, these resources are supplied and overseen by other organizations, sometimes known as cloud service providers. The services that are provided fall into many categories, such as Platform as a Service (PaaS), Software as a Service (SaaS), and Infrastructure as a Service (IaaS), all of which cater to different operating requirements. In today's modern world, cloud computing is extremely important. Cloud environments' agility, adaptability, and scalability are becoming essential as digital change gains traction for enterprises [9].

Swarm intelligence is a well-known class of metaheuristics that draw inspiration from natural occurrences. Techniques to swarm intelligence are continuous, unpredictable, and population based. Swarm intelligence has proven effective in solving a wide range of real-world issues, including the location issue in wireless sensor networks (WSNs) [10, 17], drone placement [7], robot path planning [20, 12], the network planning issue in radio frequency identification (RFID) networks [6], machine learning optimization, image processing, computer aided diagnostic systems, portfolio optimization, and more. Swarm intelligence algorithms have many uses, but

*College of Internet of Things, Jiangxi Teachers College, JTC, Yingtan, 335000, China (xiaolanlicloud@hotmail.com)
†College of Internet of Things, Jiangxi Teachers College, JTC, Yingtan, 335000, China

they have also undergone continuous modification, hybridization, and parallelization to produce the optimal outcomes.

A multiobjective task scheduling system is presented in [6] to reduce the amount of time and energy required for task scheduling. Whale Optimization Algorithm is a technique used to efficiently schedule jobs based on voltage frequency while considering variables like task execution duration and operation order. It is compared to the current PSO algorithm, is simulated on MATLAB, generates workload at random, and significantly reduces execution time and energy usage. The authors of [16] presented a task-scheduling system to manage energy usage and makespan. An algorithm known as Ant Colony Optimization is applied in a hybrid fashion to handle the parameters.

Cloud computing's explosive growth has brought opportunities and difficulties for resource scheduling optimization that maximizes efficiency and minimizes costs. With an emphasis on the most recent advancements and methodologies, this paper provides a thorough review of resource scheduling optimization techniques in cloud computing systems. First, we review the basic ideas of resource scheduling, such as work allocation, load balancing, and resource sharing. Application planning is essential because one of the main issues in cloud computing is the execution-time mappings of the task scheduler between future demand and cloud resources. To manage the diverse workload, reduce makespan, and optimize resource usage, an efficient scheduling system is required, which will improve performance metrics.

Our solution to these problems is the Hybrid Whale Optimization Algorithm (H-WOA), a multi-objective task scheduling algorithm inspired by nature that can schedule tasks in real time depending on cloud resource availability and workload demands in the future. Additionally, the suggested approach distributes resources based on end users' budgets and work priorities. The workload for the H-WOA approach is built by using datasets with different task densities and workload records from NASA and HPC2N parallel workload repositories, using the Cloudsim toolbox. Comprehensive test outcomes show that the suggested H-WOA method surpasses alternative baseline policies and enhances important factors.

The main contribution of the proposed algorithm is given below:

1. This study presents a new multi-objective work scheduling system that draws inspiration from nature. The H-WOA is intended to dynamically optimize scheduling choices in response to the workload needs that are approaching and the real-time availability of cloud resources.
2. The novel resource distribution method included in the suggested H-WOA algorithm ensures that resource allocation is in line with end-user budgets and task priorities.
3. This feature makes the scheduling system more useful and applicable to real-world cloud computing scenarios where operational and budgetary constraints are important factors to consider.
4. The study provides comprehensive experimental results that substantiate the H-WOA algorithm's efficacy. The analysis shows a considerable improvement in key performance measures when compared to baseline policy.

The rest of our research article is written as follows: Section 2 discusses the related work on Resource Scheduling Optimization, Cloud Environment. Section 3 shows the algorithm process and general working methodology of proposed work. Section 4 evaluates the implementation and results of the proposed method. Section 5 concludes the work and discusses the result evaluation.

2. Related Works. An energy-efficient task scheduling method was developed in [11] that considers varied workloads and divides tasks into two stages. A genetic algorithm is employed as a scheduling process implementation technology. It was executed in two stages. The first phase was created without taking deadline constraints into account, with tasks mapped onto virtual resources. Using task reassignment strategy based on task priority tasks mapped to virtual machines (VMs) in the second phase [14]. It was incorporated into the Amazon Cloud platform. It was contrasted with baseline methods such as SJF, FCFS, and GA variants. Eventually, based on the data, it was determined that the suggested strategy uses less energy and accurately meets task deadlines.

An energy-saving scheduling technique was put forth by the authors in [9] to address energy usage in a heterogeneous cloud environment. This scheduling system was developed using a methodology based on vacation queuing theory. To reduce the amount of time tasks spend waiting in queues, they have scheduled tasks so that when virtual machines (VMs) are busy, they must go into a queue where they can sleep [18].

When resources become available, they can then resume their work in the active queue, which reduces both the amount of time tasks spend waiting in the queue and the amount of energy they use. The simulation was done using MATLAB. It was tested against Min-Min and TSAST, and the results indicated that it had an advantage over the comparative methods for the specified metrics [3].

The technique of mapping and scheduling jobs or tasks from a scientific workflow onto distributed computing resources to accomplish goals, including limiting execution time or cost, is known as scientific workflow scheduling, or SWFS [5, 1]. Scientific processes are arranged, systematic sets of calculations intended to accomplish specific scientific objectives. These activities could include simulations, data processing, or intricate analysis involving several computational processes [2]. Because distributed computing resources are heterogeneous, with different processor kinds, memory capacities, and network bandwidths, SWFS seeks to efficiently divide these workloads among the resources at hand. Managing task dependencies, balancing numerous objectives, overcoming resource failures or uncertainties, and considering the dynamic nature of tasks and resources are some of the difficulties associated with SWFS [15, 19].

Resource scheduling methods have come a long way, yet there are still some holes in the literature. Many traditional scheduling techniques frequently ignore other important variables like energy usage and migration time in favour of maximizing makespan and resource utilization. The importance of these aspects is growing as cloud computing systems aim for increased sustainability and efficiency. Moreover, the majority of current algorithms are not adaptable enough to make decisions about scheduling that take workload demands and resource availability into account in real-time. Studies on the distribution of resources according to task priorities and user budgets are similarly few, despite the fact that these factors are critical for maximizing cloud service performance from both a technical and financial standpoint.

3. Proposed Methodology. This study introduces and evaluates the Hybrid Whale Optimization Algorithm (H-WOA), which aims to enhance resource allocation in cloud computing environments. Several key components of our proposed method are designed to assess the performance and effectiveness of H-WOA relative to traditional scheduling methods. Inspired by nature, the H-WOA is going to be developed as a multi-objective scheduling system. This strategy will maximize a number of scheduling parameters, including makespan, resource utilization, energy usage, and migration time. The flexibility to decide on runtime scheduling in response to workload requirements and real-time resource availability will be a key feature of H-WOA. To improve resource scheduling in cloud computing environments, the Hybridize Whale Optimization Algorithm (H-WOA) is being introduced and assessed in this research. A number of crucial elements in our suggested technique are intended to evaluate the efficacy and efficiency of H-WOA in comparison to conventional scheduling algorithms. The H-WOA will be created as a multi-objective scheduling system with inspiration from nature. Several aspects of scheduling, such as makespan, resource use, usage of energy, and migrate time, will be optimized by this approach. One of the main components of H-WOA will be the capacity to decide on runtime scheduling in response to workload needs and real-time resource availability. In figure 3.1 shows the architecture of proposed method.

Budgetary concerns and resource allocation based on priorities are also included in this method to guarantee a well-rounded and economical scheduling solution. The suggested method's architecture is depicted in Figure 1, whereby workload prediction models, dynamic resource monitoring, and adaptive task prioritization mechanisms are seamlessly integrated. To ensure H-WOA is reliable and flexible, its performance will be put to the test in a variety of workload conditions. Furthermore, the algorithm will be evaluated by benchmarking it against industry-standard scheduling strategies in order to show its superiority in a number of important performance parameters. To support the efficacy of the suggested approach, a thorough comparative analysis of the simulation results will be provided.

3.1. Problem Formulation. This section begins with a clearly stated problem and moves on to a discussion of the suggested system design. Assuming for the moment that there are k tasks, which are represented as $ta_k = \{ta_1, ta_2, ta_3, \ldots.., ta_k\}$, n number of virtual machines (VMs) $v_n = \{v_1, v_2, v_3, \ldots.., v_n\}$, i amount of physical hosts $h_i = \{h_1, h_2, h_3, \ldots.., h_i\}$, and lastly we have considered j datacenters, i.e. $d_j = \{d_1, d_2, d_3, \ldots.., d_j\}$. These k jobs must be meticulously scheduled or mapped onto n virtual machines (VMs), each of which is housed in i physical hosts and j datacenters. The assignment or scheduling of these jobs must minimize makespan, migration time, and consumption of energy while considering the priorities of the tasks and VMs based on the

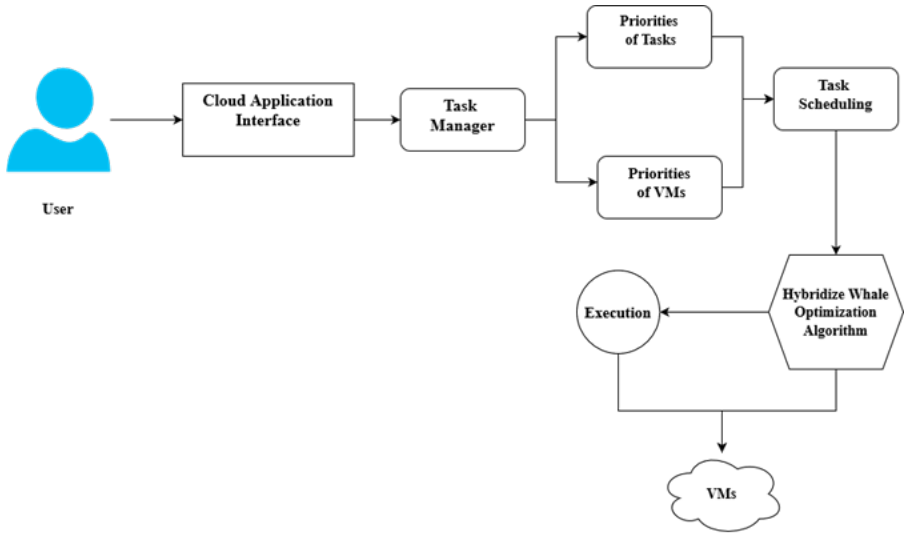

Fig. 3.1: Architecture of Proposed Method

cost of electricity per unit.

Initially, users upload tasks to the cloud console, and a broker submits them on their behalf. Brokers send these jobs to task manager, which checks if they're granted permission to use virtual resources before assigning them to scheduler. Should they own virtual resources, it is imperative to verify the service level agreed upon by the cloud provider and the user regarding the specific resources to which they have accessibility. Both the cloud service supplier and the cloud user should gain from a Service Level Agreement (SLA) that forces the service owner to implement effective multi-objective scheduling algorithms, like makespan, to reduce processing cost, and so on. In this case, the customers benefit by finishing their jobs more quickly, while the cloud owner makes more money by offering a variety of cloud services to a larger number of clients using these multi-objective scheduling tactics. The scheduler in the cloud computing standard architecture will typically assign the matching user request if the SLA between the user and the cloud provider is valid. Here, we are introducing a method in our architecture to determine tasks and virtual machine priorities depending on the cost of electricity units for datacenters. job priorities are determined because not every job has a comparable size, processing speed, or duration capability.

Task priorities are determined by the task's length and processing capacity. As a result, we must first compute the total load operating on the virtual machines to determine the priorities for the jobs. whole load on virtual machines computed using the following Eq.(3.1)

$$I_{vm} = \sum l_n \tag{3.1}$$

where l_n is the current load across all n virtual machines (VMs).

Every VM's overall load has been computed. since all virtual machines (VMs) are housed on physical hosts, for the purpose of calculating the total load on those machines. Next, we computed the total workload on the physical hosts utilizing the formula below, Equation(3.2).

$$l_h = l_{vm} / \sum h_i \tag{3.2}$$

While h_i is the total number of physical hosts considered in our model, l_h denotes the load on all physical hosts, and l_{vm} denotes the overall load on all VMs.

When many users request virtual resources from various sources, load balancers are required in the cloud computing paradigm. If a VM is unable to handle the volume of requests, a load balancer will transfer user

requests or tasks to the next virtual machine (VM) in the same pool or to a different pool. Setting a threshold value is necessary to balance the load. The performance of the cloud will be negatively impacted by static threshold values, which make putting one up in cloud computing a difficult operation. Now of the SLA, the threshold value is established based on several factors, including the volume of work, future requests, resource capacity, and so on.

Therefore, to shift load to the following virtual machine or pool, we have taken into consideration the changing threshold value. On the physical hosts, the dynamic threshold (thr_n) value is regarded as an average load.

$$thr_n = \frac{\sum_{i=1}^{n} lh_i}{n} \tag{3.3}$$

where n is the number of physical hosts and $lh_i$is the load at each physical host. The dynamic demands on the actual machine cause periodic changes in the dynamic threshold value. We determine physical hosting underutilization, overutilization, or balancing situations using this threshold.

3.2. Task Scheduling using Hybridize Whale Optimization Algorithm (H-WOA). We attempted to enhance the initial WOA execution by tackling the exploitation–exploration trade-off adjustments, keeping in mind that the two most crucial mechanisms of any swarm algorithm are exploitation (intensification) and exploration (diversification), and that the efficiency of a swarm algorithm relies strongly on the adjusted equilibrium between both steps in terms of both convergence speed and solution quality.

Modest and/or substantial enhancement tactics can be applied to any swarm algorithm to make it better. Changes to a few search equation components and adjustments to the algorithm's control variable behavior (several studies add dynamic parameter behavior) are examples of small enhancements. The term "major improvements" typically refers to combining one heuristic or metaheuristic with another. By substituting the shortcomings of one strategy with the merits (strengths) of another, hybrid algorithms integrate the best elements of two or more techniques. It is evident from looking through the literature that hybrid algorithms can be quite effective in solving various kinds of issues.

We created and put into effect a hybridized WOA method that addresses the shortcomings of the original version, building on our previous work with hybrid algorithms for swarm intelligence.

3.2.1. Encircling Prey. The purpose of modeling such actions is to replicate the algorithm's exploration and exploitation stages, in which potential solutions converged to the most optimal solution thus far. This behavior's mathematical model is:

$$\overrightarrow{D} = |\overrightarrow{C}.\overrightarrow{X} * (t) - \overrightarrow{X}(t)| \tag{3.4}$$

$$\overrightarrow{X}(t+1) = \overrightarrow{X} * (t) - \overrightarrow{A}.\overrightarrow{D} \tag{3.5}$$

The best solution so far is represented by $\overrightarrow{X}*(t)$, a solution is represented by $\overrightarrow{X}(t)$, coefficient vectors A and C are used to adjust the search's intensity and unpredictability, and D is the distance between the best solution and the current solution.

3.2.2. Spiral Bubble-Net Attacking Mechanisms. This is a simulation of the whales' spiral approach to their prey, as modeled by:

$$\overrightarrow{D'} = |\overrightarrow{X} * (t) - \overrightarrow{X}(t)| \tag{3.6}$$

$$\overrightarrow{X}(t+1) = \overrightarrow{D'}.e^{b.l}.\cos(2\pi l) \; + \overrightarrow{X} * (t) \tag{3.7}$$

where l is a random number in [−1,1], b is a constant determining the spiral's shape, and $e^{b.l}$and $\cos(2\pi l)$ determine the logarithmic spiral route.

Large-scale simulations were run with a range of benchmark datasets that had varying task densities and workload characteristics in order to verify the efficacy of H-WOA. When compared to traditional scheduling algorithms, the findings showed a considerable improvement in makespan reduction, resource utilization, energy efficiency, and migration time. Based on the comparison analysis, H-WOA proved to be a reliable solution for real-time task scheduling in cloud computing systems, continuously outperforming baseline techniques.

Table 4.1: Simulation Configuration and Environment

Parameter	**Values**
Number of tasks	100-1000
Length of tasks	700,000
Physical host RAM	16,384 MB
Physical host storage	1,000,000
Bandwidth	10,000
Number of VMs	20
RAM of VM	512 MB
OS	Linux
Number of datacenters	2

4. Result Analysis. A full discussion of simulation and findings is covered in this section. We contrasted the ACO, GA, PSO, and CS algorithms with our H-WOA. The Cloudsim [4] simulator was used to conduct the experiment. We have considered the workload of NASA [4] and HPC2N [4] computing work logs, and we have created datasets into four categories, which we go over in the section below.

An extremely well-liked online study simulator called cloudsim [4], created at the University of Melbourne, was used to simulate this study. The simulator, which is extremely helpful for a wide range of scheduling techniques, was created entirely in Java. This simulator is installed on a laptop that has an i7 processor, 1 TB hard drive, and 16 GB of RAM. Our suggested method generates workload in two different ways. At first, we created fake datasets by distributing jobs according to uniform, normal, left-, right-, and uniform distributions.

Following that, we took into consideration parallel workload traces from HPC2N [4] and NASA [4] to correctly compute parameters and determine the effectiveness of our approach. As previously noted, datasets are constructed into four categories, which are represented by the symbols da01, da02, da03, and da04, accordingly, denoting uniform, normal, left, and right distributions that are skewed. In table 4.1 shows the simulation environment.

We have assessed makespan using workload from da01, da02, da03, da04, da05, and da06, using the configuration parameters listed in Table 4.1 above.

As the number of tasks increases, all algorithms exhibit an increase in makespan, which makes sense given that more tasks often take longer to finish. Up to about 800 jobs, the suggested H-WOA appears to outperform (lower makespan) the other algorithms; after that, it dramatically improves. Out of all the algorithms examined in this situation, the ACO algorithm has the largest makespan throughout, suggesting that it may be the least efficient. For most tasks, the performance of the GA, PSO, and CS algorithms is comparable; however, for higher task counts, CS significantly outperforms GA and PSO. In figure 4.1 shows the result of Uniform Distribution of Tasks.

It seems sense that when there are more jobs to finish, more time will be required, hence as the number of tasks increases, so does the makespan for all algorithms. The ACO algorithm increases at the steepest pace and begins as the least efficient algorithm with the longest makespan. When there are fewer tasks involved, the H-WOA has the lowest makespan; however, as the number of tasks increases, its efficiency decreases, and it becomes the second least efficient when there are 1000 tasks involved. In between the ACO and H-WOA are the GA, PSO, and CS algorithms, with CS surpassing PSO and GA at about 800 tasks. In figure 4.2 shows the result of Normal Distribution.

When comparing this "Normal Distribution" plot to the last graph you shown with a "Uniform Distribution," it indicates a different situation for task distribution. A probability distribution with more tasks concentrated around a central point and less duties as you travel away from the center may be implied by a normal distribution of tasks. awareness how the algorithms behave in real-world situations requires an awareness of how their curves differ from one another. This graph and the previous one show that the algorithms function differently under various task distributions.

Each line shows how well an algorithm performed, as determined by "Makespan (ms)," or the total amount of time needed to do a specific number of tasks. The number of tasks, which is represented by the x-axis and

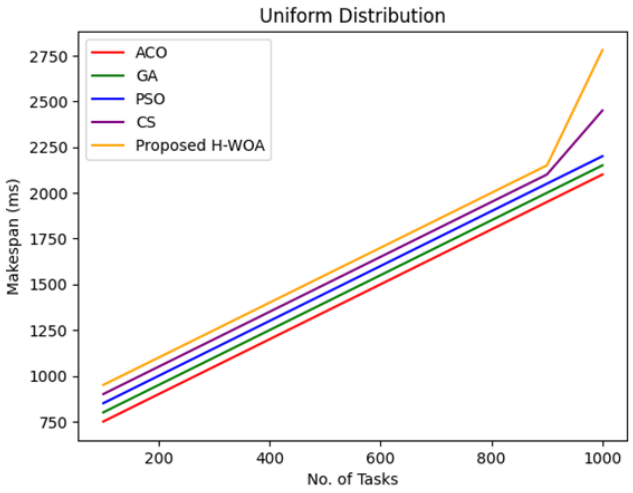

Fig. 4.1: Uniform Distribution of Tasks

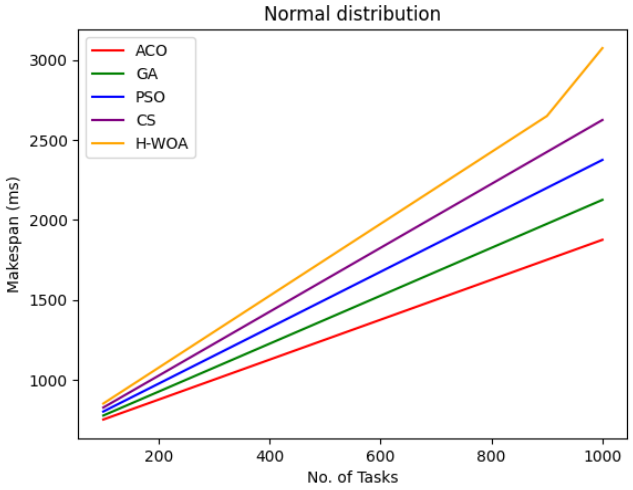

Fig. 4.2: Normal distribution of tasks

varies from 0 to 1000, is shown against the milliseconds-long makespan on the y-axis. The graph is probably used to compare how effective these algorithms are, with better performance shown by a lower makespan for a given number of tasks. As the number of tasks increases, it appears that all algorithms increase their makespan, which makes sense given that more work often requires more time. The orange-colored H-WOA method appears to outperform the other algorithms (lower makespan) at approximately 1000 tasks; nevertheless, its relative performance fluctuates along the task range. In figure 4.3 shows the result of HPC2N workload.

The "NASA Workload" graph is another line chart that bears a resemblance to the earlier ones you have presented. The makespan, or total time needed to finish a set of activities, is plotted against the number of tasks in the chart. The makespan for all algorithms grows as the number of tasks increases, indicating that larger or more complicated tasks take longer to finish. Because its line is always the lowest across the range of jobs, showing it has the shortest makespan, the H-WOA algorithm, represented by the orange line, appears to perform consistently better than the other algorithms. The GA and PSO lines exhibit a noticeable crossover

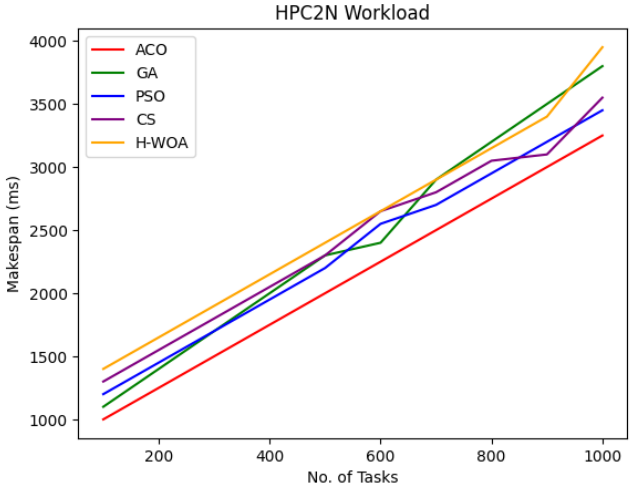

Fig. 4.3: HPC2N Workload

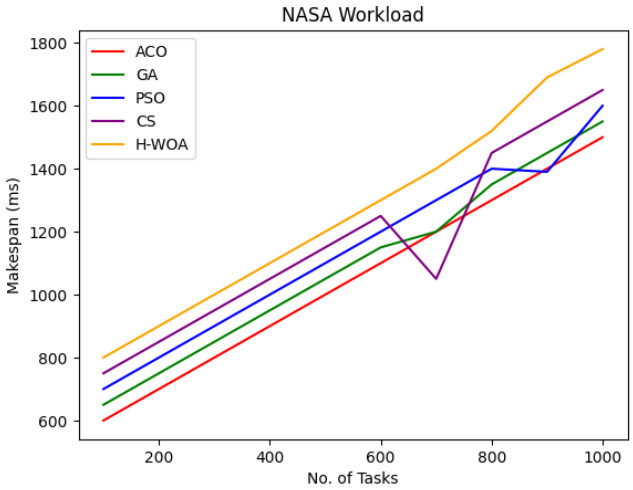

Fig. 4.4: NASA workload

point, suggesting that GA performs better than PSO for a lower number of tasks. However, PSO becomes more effective when the number of tasks goes above a specific threshold. In figure 4.4 shows the result of NASA workload.

5. Conclusion. This research has carefully examined the intricate dynamics of resource scheduling optimization tactics in cloud computing settings, showcasing important developments and approaches that tackle both established and novel problems. We found that load balancing, resource sharing, and job distribution are critically intertwined in our exploration of the fundamentals of resource scheduling. A significant development in the subject is the invention of the Hybridize Whale Optimization Algorithm (H-WOA), which provides a reliable answer to the problems associated with application scheduling. Through the integration of multi-objective optimization features, the H-WOA considers crucial elements like energy consumption and migration time, which are essential for improving cloud service performance, in addition to making makespan and resource utilization a priority. Empirical findings from tests with datasets from multiple workload repositories, includ-

ing NASA and HPC2N, and the Cloudsim toolkit confirm the superiority of the H-WOA over conventional scheduling techniques. The H-WOA guarantees optimal performance and resource efficiency by dynamically adjusting to the shifting needs of cloud resources and managing varied workloads effectively. Future research must continue to create and improve resource scheduling algorithms in order to keep up with the rapid growth of technology and the increasing complexity of cloud settings as cloud computing continues to change. As a result, the study's conclusions not only add to the body of knowledge on cloud computing optimization, but they also offer useful information that cloud service providers can use to improve service quality and operational effectiveness. In the future, scheduling techniques could potentially undergo a further revolution in resource management inside cloud computing environments with the incorporation of newer technologies like artificial intelligence and real-time data analytics.

To further enhance H-WOA's performance, future studies may investigate the integration of other meta-heuristic algorithms. To improve outcomes, combining H-WOA with Genetic Algorithms (GA) or Particle Swarm Optimization (PSO) could take advantage of each algorithm's advantages at different stages of the optimization process. By adding machine learning techniques to the H-WOA architecture, its predictive and adaptable qualities can be improved. The algorithm can make better scheduling decisions if machine learning models, for instance, are employed to forecast future workload patterns and resource availability. One area of research could be prediction models that dynamically modify H-WOA parameters depending on feedback from real-time users and historical data.

REFERENCES

[1] M. Abdullahi, M. A. Ngadi, S. I. Dishing, and S. M. Abdulhamid, *An adaptive symbiotic organisms search for constrained task scheduling in cloud computing*, Journal of ambient intelligence and humanized computing, 14 (2023), pp. 8839–8850.

[2] K. Ajmera and T. K. Tewari, *Energy-efficient virtual machine scheduling in iaas cloud environment using energy-aware green-particle swarm optimization*, International Journal of Information Technology, 15 (2023), pp. 1927–1935.

[3] D. A. Amer, G. Attiya, and I. Ziedan, *An efficient multi-objective scheduling algorithm based on spider monkey and ant colony optimization in cloud computing*, Cluster Computing, 27 (2024), pp. 1799–1819.

[4] I. Behera and S. Sobhanayak, *Task scheduling optimization in heterogeneous cloud computing environments: A hybrid ga-gwo approach*, Journal of Parallel and Distributed Computing, 183 (2024), p. 104766.

[5] A. Y. Hamed, M. K. Elnahary, F. S. Alsubaei, and H. H. El-Sayed, *Optimization task scheduling using cooperation search algorithm for heterogeneous cloud computing systems.*, Computers, Materials & Continua, 74 (2023).

[6] S. M. Kak, P. Agarwal, M. A. Alam, and F. Siddiqui, *A hybridized approach for minimizing energy in cloud computing*, Cluster Computing, 27 (2024), pp. 53–70.

[7] M. I. Khaleel, *Efficient job scheduling paradigm based on hybrid sparrow search algorithm and differential evolution optimization for heterogeneous cloud computing platforms*, Internet of Things, 22 (2023), p. 100697.

[8] M. Khenwar, A. Sisodia, S. Vishnoi, and R. Kumar, *Exploration: Cloud computing scheduling techniques*, Scandinavian Journal of Information Systems, 35 (2023), pp. 673–679.

[9] S. Mangalampalli, G. R. Karri, and M. Kumar, *Multi objective task scheduling algorithm in cloud computing using grey wolf optimization*, Cluster Computing, 26 (2023), pp. 3803–3822.

[10] R. F. Mansour, H. Alhumyani, S. A. Khalek, R. A. Saeed, and D. Gupta, *Design of cultural emperor penguin optimizer for energy-efficient resource scheduling in green cloud computing environment*, Cluster Computing, 26 (2023), pp. 575–586.

[11] H. Mikram, S. El Kafhali, and Y. Saadi, *Hepga: A new effective hybrid algorithm for scientific workflow scheduling in cloud computing environment*, Simulation Modelling Practice and Theory, 130 (2024), p. 102864.

[12] S. Pal, N. Jhanjhi, A. S. Abdulbaqi, D. Akila, F. S. Alsubaei, and A. A. Almazroi, *An intelligent task scheduling model for hybrid internet of things and cloud environment for big data applications*, Sustainability, 15 (2023), p. 5104.

[13] P. Pirozmand, H. Jalalinejad, A. A. R. Hosseinabadi, S. Mirkamali, and Y. Li, *An improved particle swarm optimization algorithm for task scheduling in cloud computing*, Journal of Ambient Intelligence and Humanized Computing, 14 (2023), pp. 4313–4327.

[14] F. S. Prity, M. H. Gazi, and K. A. Uddin, *A review of task scheduling in cloud computing based on nature-inspired optimization algorithm*, Cluster computing, 26 (2023), pp. 3037–3067.

[15] M. Raeisi-Varzaneh, O. Dakkak, A. Habbal, and B.-S. Kim, *Resource scheduling in edge computing: Architecture, taxonomy, open issues and future research directions*, IEEE Access, 11 (2023), pp. 25329–25350.

[16] Y. Wang, S. Dong, and W. Fan, *Task scheduling mechanism based on reinforcement learning in cloud computing*, Mathematics, 11 (2023), p. 3364.

[17] 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, 12 (2023), p. 8.

[18] M. Zeedan, G. Attiya, and N. El-Fishawy, *Enhanced hybrid multi-objective workflow scheduling approach based artificial bee colony in cloud computing*, Computing, 105 (2023), pp. 217–247.

[19] G. Zhou, W. Tian, R. Buyya, and K. Wu, *Growable genetic algorithm with heuristic-based local search for multi-dimensional resources scheduling of cloud computing*, Applied Soft Computing, 136 (2023), p. 110027.
[20] G. Zhou, W. Tian, R. Buyya, R. Xue, and L. Song, *Deep reinforcement learning-based methods for resource scheduling in cloud computing: A review and future directions*, Artificial Intelligence Review, 57 (2024), p. 124.

Edited by: Rajkumar Rajavel
Special issue on: Cognitive Computing for Distributed Data Processing and Decision-Making in Large-Scale Environments
Received: May 12, 2024
Accepted: Jun 9, 2024