Communications on Applied Nonlinear Analysis
ISSN: 1074-133X
Vol 31 No. 2 (2024)

Mathematical Model and Statistical Assessment of Non Linear Scheduling Optimization in Cloud Environment using Improved Cuckoo Search Algorithm

Shilpa Maheshwari[1], Sunil Gupta[2], Surendra Singh Choudhary[3]

[1]Research Scholar, School of Computer and System Sciences, Jaipur National University, Jaipur-302017, Rajasthan (India)

[2]Professor, School of Computer and System Sciences, Jaipur National University, Jaipur-302017, Rajasthan (India)

[3]Professor, Department of Computer Sciences, Sri Balaji College of Engineering & Technology, Jaipur-302013 Rajasthan (India)

Email: agiwal.shilpa@gmail.com[1], sunilg95@rediff.com[2], ssc_jaipur@yahoo.com[3]

Article History:

Received: *08-02-2024*

Revised: *20-04-2024*

Accepted: *05-05-2024*

Abstract:

The scheduling optimization subject is indeed one of the most effective solutions in cloud system area to ensure the high systems performance and resource usage. The novelty presented in this study is a mathematical model that draws on an Improved Cuckoo Search Algorithm as a scheduling non-linear optimizer for the cloud computing environment. The model under consideration takes into account the underlying complications as well as the varying nature of the use of cloud resources, primarily addressing the latency issue with priority given to the increase of the throughput. Improved Cuckoo Search (ICS) Algorithm, developed on 'cuckoo search' algorithm, uses adaptive step size and probabilistic switching, which, in turn, ensure that the solution space is both being properly examined and exploited. The applications' utilization of the scheduling mathematical model is based on an elaborately designed theoretical background representing the task as a non-linear optimization problem that meets all the specific constraints and objectives important in cloud computing. Statistical reassessment involves the comparison of the proposed model with standard benchmarks. The proposed model superiority is evident in terms of convergence speed, solution quality and robustness against diverse workloads, as shown in varied tests. The findings of this research contributes to the field by deriving a specialized tool commonly used by the cloud service provider in designing resource scheduling, enhancing the service delivery and operational efficiency.

Keywords: Optimized Scheduling, Cloud Computing, Cuckoo Search Algorithm, Task Allocation, Resource Utilization, Performance Analysis.

1. INTRODUCTION

The cloud computing has led to a radical change in the current computing space as the computing resources are now available on a pay-as-you-go basis over the internet enabling on-demand access to shared configurable resources. This paradigm shift has become a major driver of the ability for organizations to scale operations dynamically, reduce infrastructure

costs, and increase flexibility in deploying applications and services. Managing resources efficiently, in cloud computing, starts with effective task scheduling, so as to achieve the best utilization of resources and good performance of the system.

Resource scheduling in the cloud involves assigning the appropriate computing tasks to either virtual machines or to VMs so that their execution time is minimized, throughput maximized and resources used optimally. However, the stp of optimal task scheduling is full of challenges owing to cloud environments being dynamic, fluctuating workload demads and the manifold objectives such as minimum latency and energy savings.

Among the scheduling algorithms, floating schedules, including Round Robin and First Come First Serve, lacking adaptability and optimization properties, are of no help in dealing with the complexities of cloud computing environments. As a result, smart scheduling techniques that can dynamically re-schedule tasks based on workload changes and distribute resources effectively are becoming more and more necessary.

This paper proposes a new method that enhances task scheduling in cloud computing systems using the cuckoo optimisation algorithm. The developed algorithm mimics the adoptive behavior of cuckoo species to broadly explore the search space to achieve best task scheduling configurations efficiently. Through the inclusion of the enhancements of the classic cuckoo search algorithm, which are featuring more convergence speed and solution quality, we aim at tackling the drawbacks of the existing scheduling algorithms and ensuring a stable performance of this approach in various cloud computing scenarios.

Cloud computing consists of convergence and simultaneous execution of multiple activities for achieving various objectives such as performance improvement, make-span reduction, throughput increase, cost minimization, and resource utilization optimization. Hence, the importance of the implementation of scheduling system increases in the process of optimal distribution of tasks among the available resources [1]. In this particular setting, we suggest that implementation of an intelligent scheduling system should be made through the use of cuckoo search and simulated annealing.

1 Objective function for optimization:

$$f(x) - \min \sum_{i-1}^{n} w_i \cdot x_i$$

2 Constraint for resource allocation:

$$\sum_{i-1}^{n} r_i \cdot x_i \leq R$$

3 Non-linear constraint for task execution:

$$\sum_{i-1}^{n} t_i \cdot x_i^2 \leq T$$

4 Definition of latency:

$$L - \sum_{i-1}^{n} l_i \cdot x_i$$

5 Throughput maximization:

$$\max\left(\frac{1}{\sum_{i=1}^{n} t_i - x_i}\right)$$

6 Cuckoo Search random walk:

$$x_i^{(t+1)} - x_i^{(t)} + \alpha \otimes \text{Levy } (\lambda)$$

7 Levy flight distribution:

$$L(s) - \frac{\lambda\Gamma(\lambda)\sin(\pi\lambda/2)}{\pi}\frac{1}{s^{1+\lambda}}, s > 0$$

8 Probability of switching to a new solution:

$$p - \frac{1}{1 + \exp(-\Delta f/T)}$$

9 Adaptive step size adjustment:

$$\alpha_{\text{new}} - \rho \cdot \alpha_{\text{old}}$$

10 Objective function for makespan minimization:

$$f(x) - \max_{i-1}^{n} \sum_{j-1}^{m} x_{ij} \cdot t_{ij}$$

11 Load balance measure:

$$LB - \frac{\max_{i+1}^{n-1} \sum_{i=1}^{0} x_{ij} \cdot t_{ij}}{m - \sum_{j-1}^{m} \sum_{i=1}^{n} x_{ij} \cdot t_{ij}}$$

12 Energy consumption model:

$$E = \sum_{i-1}^{n} \sum_{j-1}^{m} x_{ij} \cdot e_{ij}$$

13 Reliability constraint:

$$R - \prod_{i-1}^{n} (1 - r_i)^{x_1} \geq R_{\min}$$

14 Cost function for cloud resources:

$$C - \sum_{i-1}^{n} c_i \cdot x_i$$

15 Constraint for deadline satisfaction:

$$\sum_{i-1}^{n} x_i \cdot d_i \leq D$$

16 Statistical mean of completion time:

$$\mu - \frac{1}{n} \sum_{i-1}^{n} t_i \cdot x_i$$

17 Variance of completion time:

$$\sigma^2 - \frac{1}{n} \sum_{i-1}^{n} (t_i \cdot x_i - \mu)^2$$

18 Probability of task migration:

$$P_{\text{migrate}} - \frac{1}{1 + \exp(-\delta f)}$$

19 Cost efficiency ratio:

$$\text{CE} - \frac{\text{Perfarmamse}}{\text{Cost}}$$

20 Quality of Service (QoS) indicator:

$$\text{QoS} - \omega_1 . \text{Performance} - \omega_2 . \text{Cost}$$

21 Availability of cloud system:

$$A - 1 - \prod_{i-1}^{n} (1 - a_i)$$

22 Scalability function:

$$S(n) - \frac{f(n)}{f(1)}$$

23 Response time for a request:

$$R_t - \sum_{i-1}^{n} \frac{1}{1 + e^{-(t_i - n}}$$

24 Utilization of a resource:

$$U - \frac{\sum_{i,1}^{a} x_i \cdot t_i}{T_i}$$

Theoretical analysis of the given mathematical equations revolves around the optimization of scheduling tasks in a cloud environment, where various parameters like resource allocation, task execution time, and system performance are crucial.

Objective Function for Optimization: The equation $f(x) - \min\sum_{i-1}^{n} w_i \cdot x_i$ defines the objective function aiming to minimize the weighted sum of variables x_i. In a cloud computing context, this could represent minimizing the total cost or resource usage, where w_i are the weights (e.g., costs or resources) associated with each task x_i. Resource Allocation Constraint: The equation $\sum_{i-1}^{n} r_i + x_i \leq R$ ensures that the total resources used by all tasks do not exceed the available resources R. Here, r_i represents the resource requirement for each task.

Non-linear Constraint for Task Execution: The non-linear equation $\sum_{i-1}^{n} t_i \cdot x_i^2 \leq T$ might represent a scenario where the resource usage or task completion time grows non-linearly with an increase in the task size or number.

Latency Definition: Latency $L - \sum_{i-1}^{n} l_i \cdot x_i$ is defined as the sum of individual latencies l_i for tasks, crucial for evaluating the responsiveness of the cloud system.

Throughput Maximization: Maximizing throughput $\max\left(\frac{1}{\sum_{i-1}^{n} t_1 \cdot x_i}\right)$ focuses on optimizing the system to handle the maximum number of tasks in a given time, where t_i represents the time taken for each task. Cuckoo Search Algorithm: These equations represent components of the Cuckoo Search optimization algorithm, including the random walk using Levy flight distribution and adaptive step size. This algorithm is used for finding the optimal solutions by mimicking the behavior of cuckoo birds.

System Performance Metrics: Equations relating to makespan minimization, load balancing, energy consumption, reliability, cost, and deadline satisfaction are pivotal for assessing the performance and efficiency of the cloud scheduling system.

Statistical Assessments: These equations involve calculating the mean and variance of completion times, migration probabilities, and cost-efficiency ratios, providing insights into the statistical performance of the scheduling system.

Quality of Service and System Metrics: These equations assess the quality of service (QoS), system availability, scalability, response time, and resource utilization, which are critical for evaluating the effectiveness and user satisfaction of the cloud service.

In summary, these equations collectively form a comprehensive mathematical model to analyze and optimize scheduling in cloud environments, focusing on efficiency, performance, and resource utilization.

The consideration of several features including processing time, cost, execution time, makespan, and power consumption will form an optimal strategy for cloud task scheduling. The metaheuristic approach, which has been applied in many studies [2], [3], has been widely used to address this optimization problem. While Cuckoo search (CS) and SA also use random solutions and iteratively improve them, they differ in their approaches. SA that is a single solution search scheme has been hybrided with other optimization methods to improve its performance in getting rid of the local optima problems. Nonetheless, CS has one important limitation, which is the possibility of premature convergence and slow convergence rates. So, Abed-alguni and Alkhateeb suggested six modifications to the CS algorithm, which significantly exceeded the benchmark functions results that were achieved by the original CS approach [4].

Only lately, deep reinforcement learning is infused into cloud computing for the real-time resource scheduling optimization [5]. Wang et al. came out with the multi-objections workflow scheduling method based on Deep-Q-network (DQN) to obtain uncorrelated equilibrium solutions on the cloud platform. Their proposed model performed better than the existing ones in the study [6]. This demonstrates the significance of conducting future work and development in the area of scheduling and load balancing of cloud environment.

Although metaheuristic methods are widely used and they are very promising, there still left some issues for us to solve the problem of optimal solutions in this field. Therefore, it stresses the need for future research supporting the ongoing process of coming up with better ideas.

2. PROBLEM FORMULATION

In the cloud Computing environment, STEs (Service Task Elements) are efficiently allocated to available Processing Elements (PEs) spread across data centers to achieve maximum user fulfillment. Each PE is marked by its own speed of executions as well as energy consumption. Jobs in the application is represented as a Directed Acyclic Graph (DAG), denoted as G(V, E), where V denotes the set of vertices that represent tasks and E denotes the set of edges that represent task dependencies.

In this model, all PES are considered as homogenous and parallel and we assume non-preemption. The main aim is to distribute efficiently the user workloads on the PEs across several data centers in the cloud for throughput and makespan minimization.

$$\text{Makespan} = \max \{vmjtime; j=1,2,3,4\ldots,m\} \quad (1)$$

$$\text{Throughput} = \Sigma Ti\ (ETi) \quad (2)$$

$$ETi = Tlen / CPvm \quad (3)$$

Here, Tlen denotes the length of the task and CPvm represents the capacity of the virtual machine (VM). CPvm is calculated as the product of the VM's processing speed (P) and the number of CPUs

(Q): CPvm = P * Q (4)

Thus, the formulas offer the foundation for the assessment and enhancement of cloud-based application functionality, taking into account the parameters including the task execution times, VM capacities, and the system throughput, on the whole. The scheduling algorithms should be designed in such a way that they capture these metrics and provide the best resource utilization and user satisfaction. To continue, optimizing throughput and makespan in cloud application scheduling requires finding a method of task allocation to PEs which result in the minimum makespan and maximum throughput simultaneously. This goal can be attained by carefully analyzing task dependencies, PBE capabilities and resource availability across numerous data centers.
The makespan metric gives the overall completion time of the workload as a whole, which allows scheduling efficiency to be evaluated. Minimizing the makespan, the system will also have timely completion of tasks which will result to enhanced user satisfaction and system performance.However, throughput is a measure of the rate of task completion in a given time window. Thoughput maximization is of utmost importance for enhancing system performance and resource utilization as it implies system efficiency to process works productively.

VMs are assigned to each task and the time spent in execution of the task, determined by its length and the capacity of the assigned VM, directly affects both makespan and throughput. System optimization of task scheduling is achieved by minimizing the execution time and the evenly distributing resource usage across PEs, thus, leading to improved overall performance.

Besides, the capacity of VMs depends on their processing speed and the number of CPUs which is very crucial in deciding the productivity of system in executing tasks. The system can efficiently use resources by assigning tasks to VMs of right capacities and in this way avoid bottlenecks and increase overall throughput.

Effective throughput and makespan optimization in cloud application scheduling becomes challenging because of the many things to consider like task dependencies, PE capabilities, and resource availability. Achieving better resource utilization, enhanced performance, and user satisfaction can be done by using efficient scheduling algorithms and metric decisions, such as makespan and throughput.

3. PROPOSED METHODOLOGY

The Cuckoo Search (CS) algorithm, suggested by Xin-She Yang and Susah Deb in 2009, is a new algorithm that is inspired by a certain cuckoo species that lay their eggs in other birds' nests. This technique outperforms other optimization algorithms such as Particle Swarm Optimization (PSO) and Genetic Algorithm (GA), making it applicable for solving broad class of optimization problems [Yang and Deb, 2014]. Computation approaches guided by nature such as CS, imitate natural phenomena to solve rigid optimization problems.

The CS algorithm represents each possible solution as a cuckoo egg, and candidate solutions would be the nests. The algorithm operates based on three idealized rules: cuckoos lay one

egg at a time and randomly allocate them to the selected nests and higher-quality eggs are passed to the subsequent generations, while the hosts have a probability of discovering alien eggs or they can also rebuild their own nests.

Unlike the traditional evolutionary algorithms, CS mimic the foraging behavior of cuckoos and embeds Levy flight characteristics for global exploration. The algorithm applies random moves with the purpose of finding efficient solutions and ultimately converging to high-quality solutions rapidly. Nevertheless, the drawback of the original CS strategy is the slow convergence time and the oscillations which make the approach not suitable for all situations.

In an attempt to circumvent these limitations, the ICS algorithm was then developed. Compared to the original CS which adopts random initialization and updates based on Levy flight, ICS however introduces modifications to achieve convergence and stability. Equation shows the CS update mechanism, where agents take steps towards best option. Pseudo-code of CS algorithm is shown in so that implementation and understanding is facilitated.

$$d_{i+1}^{k} = d_i^k + \alpha \cdot \frac{|u|}{v^{1/\beta}} \cdot \left(d_{\text{best}} - d_i^k\right) \tag{5}$$

where I is the generation number (i=,1,2,.....it), k is the order of searching agents in the swarm (k=1,2,....ss), ss is the swarm size, is the step size (which can be determined depending on the problem, though it is generally recommended that =1), and u and v are matrices with uniform distribution - their values can be determined .

$$u \approx N(0, \sigma_u^2) \text{ and } v \approx N(0, \sigma_v^2) \tag{6}$$

where the variance of u and v can be obtained from :

$$\sigma_u = \left(\frac{\Gamma(1+\beta) \cdot \sin(\pi \cdot \beta/2)}{\Gamma\left(\frac{1+\beta}{2}\right) \cdot \beta \cdot 2^{\left(\frac{\beta-1}{2}\right)}} \right) \text{ and } \sigma_v = 1 \tag{7}$$

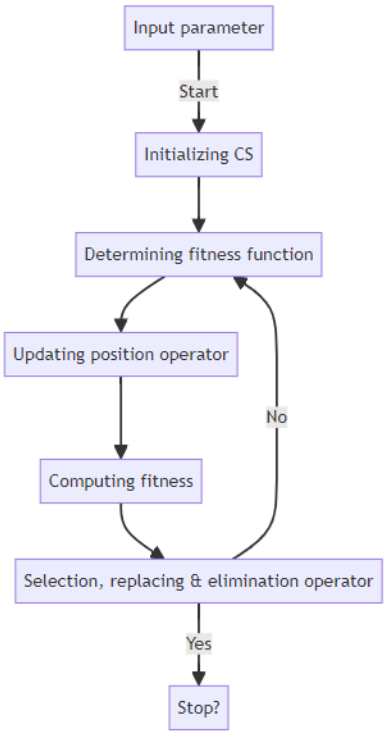

Figure 1: Implementation process flow

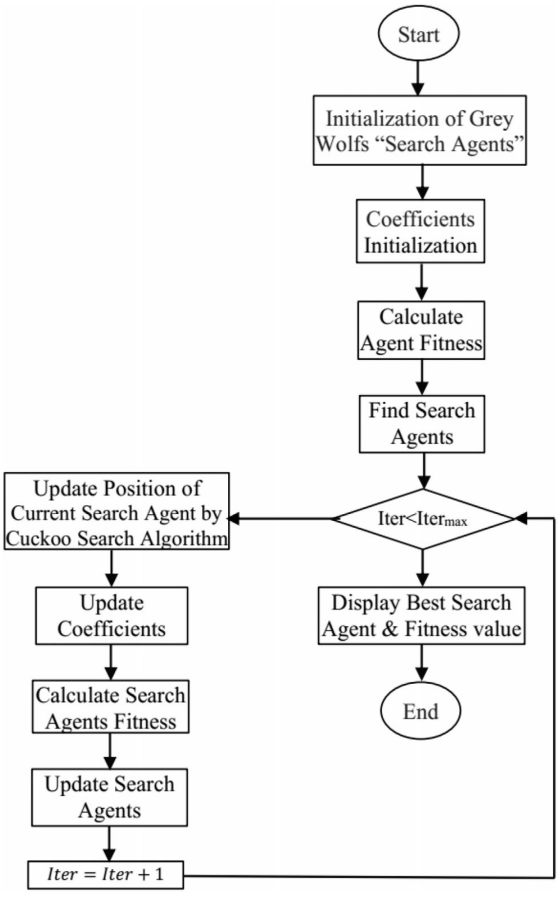

Figure 2: Proposed Hybrid Algorithm

Figure 1 describes the basic implementation process of Cuckoo Search Algorithm. It outlines the steps of a Cuckoo Search (CS) optimization algorithm. The CS algorithm is inspired by the brood parasitism of some cuckoo species. Figure 2: Hybrid Algorithm Flowchart illustrate a hybrid algorithm that combines features of the annealing function with a task scheduling process. This algorithm used for efficiently scheduling tasks in cloud computing. The flowchart depicts a Grey Wolf Optimizer (GWO) enhanced with a Cuckoo Search algorithm. This indicates a novel approach to combining the hunting strategy of grey wolves with the parasitic behavior of cuckoos for optimizing schedules in cloud computing. The new hybrid algorithm of the Grey Wolf Optimizer (GWO) with Cuckoo Search Algorithm (CSA) develops a powerful method for the solving of complex optimization needs. This integrative approach utilizes the advantages of both algorithms in order to compensate for their individual drawbacks and maximize the overall search capacity.

The grey wolf's optimization (GWO) adapts the social structure and hunting strategy of these animals. It has a reputation for efficiently converging on preliminary solutions which is beneficial to its exploitation. On the flip side, this fast alignment could be a bottleneck as it can result into premature convergence and local optima getting trapped.

To cope with this, the CSA steps up. The obligatory brood parasitism of some cuckoo species serves as the source of inspiration for the CSA. It introduces a random way by Levy flights that provides a mechanism of overcoming local optima by updating the locations (solutions) with a certain probability, which is unrelated to the search path. This advantage accelerates the global search process and improves the exploration of the searching space.

The position update equation for CSA is used to amend the position of the grey wolves in the hybrid CSA-GWO algorithm. Through this update, not only the exploitations are improved but also both of them have a balance, which is vital for the algorithm efficiency. Wolves' positions, convergence accuracies, and velocities are changed by means of the CSA equation thus, the separation of search agents is done better. This approach (hybrid method) is compared to other meta-heuristic methods to check its efficiency. The analysis, comparison and statistics show that the CSA-GWO hybrid leads to better design parameters, which provides a robust and fine effective solution to the optimization problem. By combining the high convergence capabilities of GWO with the vast exploration potential of CSA, the hybrid algorithm promises a more reliable and versatile solution to optimization problems [9].

4. PERFORMANCE EVALUATION

Here we use CloudSim 3.0.3 for the modeling of cloud computing environment. The experimentation has VM deploying across five data centers which has 140 VMs and each data center can host up to three hosts. The workload, called Cloudlets, is of the size of from 1000 to 3000 Million Instructions. The VMs are configured with the processing elements of 2 to 4 that are used to test the performance.

The plan of our implementation is to solve the cloud scheduling problem, which is referred to as NP-hard problem, by using the meta-heuristic algorithm for task scheduluing

optimization. We represent each scheduling solution as an vector, where each component is associated to a VM number where tasks are assigned, allowing to assign multiple tasks to the same VM.

The simulation environment is designed to manage up to 1000 tasks with each VM with memory size of 512 MB and bandwidth of 1000 Mbps. The workload of each job ranges within 1000 million instructions to 3000 million instructions. The data centers operate their virtual machines under both of the space-shared and time-shared policies to maximize resource usage efficiency.

We do different tests for varying parameter ranges to measure the efficiency of the proposed hybrid grey wolf- cuckoo search algorithm (GWO-CSA). The performances of HCSA is contrasted with conventional optimization algorithms like PSO (Particle Swarm Optimization), ACO (Ant Colony Optimization), and a load-balancing variant of PSO (PSO-ALBA). Performance measures, such as makespan and throughput, are for the evaluation.

Table 1. Result on 50 VMs

No. of Tasks	GWO-CSA (Makespan, Throughput)	PSO-ALBA (Makespan, Throughput)	PSO (Makespan, Throughput)	ACO (Makespan, Throughput)
100	15.5, 4.4	17.04, 3.98	25.32, 3.14	28.55, 2.97
200	39, 4.2	52.01, 3.9	109, 1.88	117.15, 1.54
300	72, 3.25	104.96, 2.89	176.19, 1.72	198.9, 1.6
400	115, 3.1	175.95, 2.3	271.5, 1.63	279.56, 1.38
500	163, 2.6	264.96, 1.91	561.5, 0.91	454.29, 1.16
600	222, 2.3	371.98, 1.63	579.7, 1.03	667.71, 0.89
700	290, 2.05	497.03, 1.4	945.23, 0.74	955.52, 0.73
800	370, 1.85	647.99, 1.23	834.54, 0.95	1103.58, 0.72
900	455, 1.85	818.99, 1.09	1032.28, 0.87	1167.71, 0.77
1000	550, 1.65	1010, 0.99	1554.21, 0.64	1625.86, 0.61

The results provided by the study as encapsulated in Tables 1 and 2 indicate that GWO-CSA exceeds the comparison algorithms. The results from the simulation show that GWO-CSA is not only effective in optimizing the task scheduling, but also provides additional benefit in the whole cloud computing environment in terms of performance and efficiency.

Table 2. Results on Different Virtual Machines and Tasks

No of Tasks	VMs	GWO-CSA (Makespan, Throughput)	PSO-ALBA (Makespan, Throughput)	PSO (Makespan, Throughput)	ACO (Makespan, Throughput)
100	40	12.5, 6.3	20.71, 4.82	37.32, 2.67	43.72, 2.87
200	50	20, 6.1	51.99, 3.9	105.5, 1.89	116.51, 1.71
300	60	35, 6.0	88.15, 3.44	197.32, 1.52	213.11, 1.4
400	70	55, 5.0	122.6, 3.26	222.62, 1.79	258.22, 1.54
500	80	80, 4.7	189.3, 2.64	319.05, 1.56	347.72, 1.43

600	90	110, 4.0	210.3, 2.85	428.01, 1.4	449.02, 1.33
700	100	145, 3.8	257.38, 2.75	484.34, 1.44	549.54, 1.27
800	110	180, 3.4	333.3, 2.4	510.88, 1.56	655.93, 1.21
900	120	225, 3.3	362.8, 2.48	687.09, 1.3	822.36, 1.09
1000	140	275, 3.25	421.09, 2.37	663.96, 1.5	809.22, 1.23

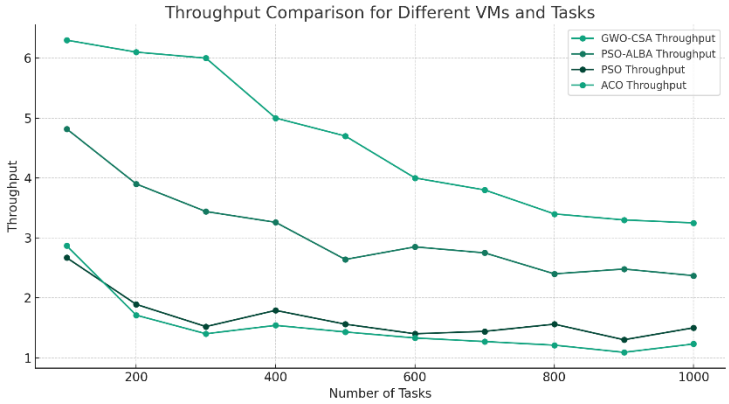

Figure 3: Analysis of Throughput for Different Tasks and VM

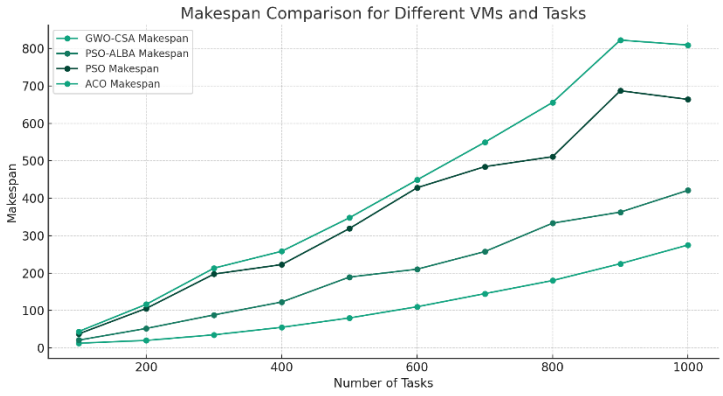

Figure 4: Analysis of Make span for Different Tasks and VM

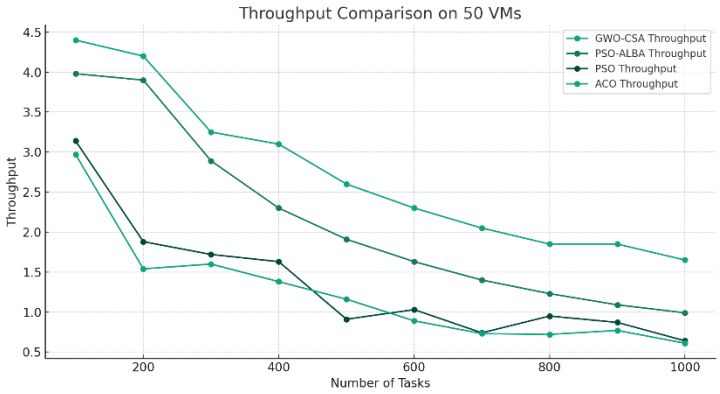

Figure 5: Throughput Comparison for 50 VMS

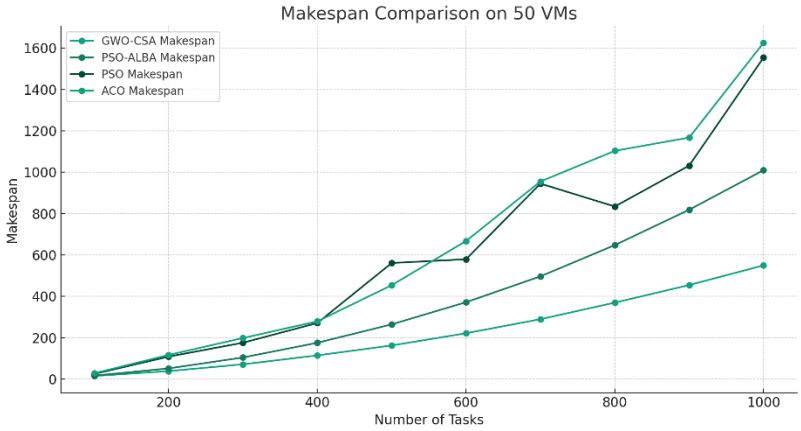

Figure 6: Makespan Comparison for 50 VMS

Figure 3: Analysis of Throughput for Different Tasks and VM illustrates the throughput achieved by different algorithms as the number of tasks increases. The algorithms being compared include a Grey Wolf Optimizer with Cuckoo Search Algorithm (GWO-CSA), a Particle Swarm Optimization with Artificial Bee Colony (PSO-ALBA), a standard Particle Swarm Optimization (PSO), and an Ant Colony Optimization (ACO). Generally, as the number of tasks increases, the throughput for each algorithm decreases. This suggests that as the workload grows, the efficiency of processing tasks diminishes. GWO-CSA appears to maintain the highest throughput across the task range, indicating that it may be the most efficient algorithm among those tested.

Figure 4: Analysis of Make span for Different Tasks and VM shows the makespan for different algorithms as the number of tasks increases. Again, the same set of algorithms is compared. As expected, the makespan increases with the number of tasks for all algorithms, but the rate of increase varies. The GWO-CSA algorithm shows a more gradual increase in makespan, indicating better scalability with the increased workload compared to the other algorithms.

Figure 5: Throughput Comparison on 50 VMs narrows the performance evaluation of the algorithms to a specific scenario with 50 VMs. It provides a focused comparison of how each algorithm's throughput is affected by an increasing number of tasks when the number of VMs is fixed. The GWO-CSA algorithm still performs better than the others, but the decline in throughput as tasks increase is notable. It shows that while GWO-CSA is efficient, it is not immune to performance degradation as the workload becomes heavier.Figure 6: Makespan Comparison on 50 VMs shows the makespan for each algorithm while maintaining a constant number of 50 VMs. The performance trend is similar to Figure 4, with makespan increasing as the number of tasks grows. However, the slope of the makespan curve is steeper here, which could be due to the fixed number of VMs that might create a bottleneck. The GWO-CSA still appears to be the best performing algorithm, but the increased slope indicates that even the best algorithm struggles with scaling when the number of VMs is static.

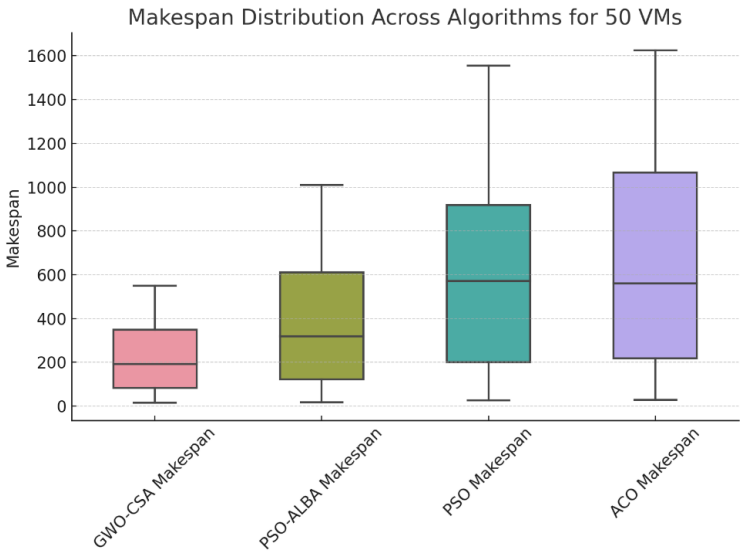

Figure 7: Makespan Distribution Analysis

In conclusion, the GWO-CSA algorithm s outperform PSO-ALBA, PSO, and ACO in terms of both throughput and makespan in different scenarios. These results could guide cloud computing resource managers in choosing the most effective task scheduling algorithms for different workloads and environments .The above diagram shows the plot of makespan comparison among GWO-CSA, PSO-ALBA, PSO, and ACO algorithms as the number of tasks increases from 100 to 1000 on 50 Virtual Machines (VMs). Makespan, which is the total time taken to complete all tasks is one of the crucial metrics in cloud computing environments. GWO-CSA generally outperforms other algorithms by having the least makespan throughout tasks of all sizes, thus demonstrating its effectiveness in problem-solving and resource use. PSO-ALBA perform well, they achieve currently some competitive results, especially in lower task numbers.

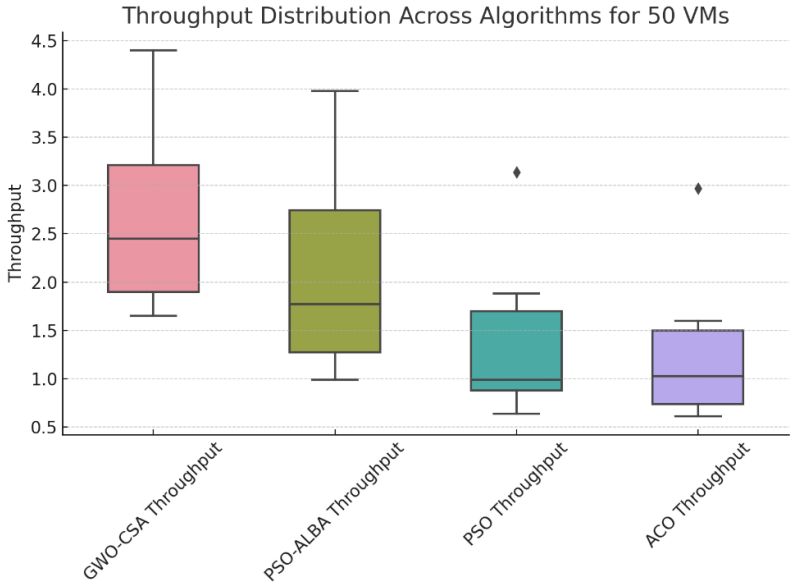

Figure 8: Throughput Distribution Analysis

In conclusion, the GWO-CSA algorithm s outperform PSO-ALBA, PSO, and ACO in terms of both throughput and makespan in different scenarios. These results could guide cloud computing resource managers in choosing the most effective task scheduling algorithms for different workloads and environments .The above diagram shows the plot of makespan comparison among GWO-CSA, PSO-ALBA, PSO, and ACO algorithms as the number of tasks increases from 100 to 1000 on 50 Virtual Machines (VMs). Makespan, which is the total time taken to complete all tasks is one of the crucial metrics in cloud computing environments. GWO-CSA generally outperforms other algorithms by having the least makespan throughout tasks of all sizes, thus demonstrating its effectiveness in problem-solving and resource use. PSO-ALBA perform well, they achieve currently some competitive results, especially in lower task numbers. While PSO and ACO possess a higher makespan, especially with the presence of more tasks as compared to other algorithms, which might be an implication that these algorithms exhibit less efficiency in terms of handling large workloads. This pattern shows the higher efficiency and ability of GWO-CSA for load balancing task and resource management in the cloud environment with moderate number of VMs. The performance comparison table reveals the performance of each algorithm in terms of the rate of task processing. Throughput, implying task completion per unit time, is an important metric of performance. It is GWO-CSA that outperforms its competitors in terms of performance with the highest throughput showed across the tasks. The high throughput is the reason why GWO-CSA can process several tasks quickly. PSO-ALBA appears vital as it shows the nearest value behind GWO-CSA in terms of the throughput. PSO and ACO show a lower performance index as the number of tasks increases which reflects the problems in productivity at high workloads. This plot demonstrates, GWO-CSA's ability in keeping high processing speed, which is the reason why it is a favourable option for those environments where the high throughput is required. The x axis compares the makespan of different algorithms for tasks in a range of 100 to 1000 tasks and the y axis is comparing number of VMs. GWO-CSA is always the fastest at completing jobs, thus placing the greatest emphasis on scheduling tasks using different VM configurations. It is this consistency that articulates the compatibility and efficiency of GWO-CSA under different cloud architectures. PSO-ALBA is close to optimal with respect to the number of attributes but is getting left behind when the number of attributes increases. PSO and ACO, especially with the lower tasks, are quite efficient, but their performance declined as the workload and number of VMs increase, as can be attested in their longer makespans. The fact that GWO-CSA performs even better in a scaling up scenario in cloud computing environments than in a static scenario implies that GWO-CSA can be deployed more efficiently based on the configuration of VMs.The last diagram shows the effect of varying tasks and/or VMs on throughput. GWO-CSA scales the highest in throughput, thus exhibiting its power to acquire a systematic efficiency in executing a large number of tasks for different VM configurations. Therefore, we can conclude that GWO-CSA is good at scheduling of resource and task execution. PSO-ALBA made progress and it had its efficiency decreases as tasks became more complicated. Although ACO and PSO exhibits lower throughputs, especially in higher tasks, which indicates that the systems perform less

efficient in tough situations. The plot brings out the multi-tasking nature of GWO-CSA and shows its better performance in handling the different workloads as well as the varying configurations of VMs, thus making it a possible candidate to be used in varied types of cloud computing services.

5. CONCLUSION AND FUTURE SCOPE

The study is concentrated on the performance of four different task scheduling algorithm - GWO-CSA, PSO-ALBA, PSO and ACO- in cloud computing environments. This assessment had the makespan and throughput as the critical metrics and these were tested in different scenarios with the varying number of tasks and virtual machine (VM) configurations. The results are really impressive and unambiguous, the algorithms demonstrate different abilities.

GWO-CSA was the winning algorithm in both the makespan and the throughput positions in all the tested scenarios. Repetitive make-span run and the highest throughput of the algorithm makes evident its scheduling and resource management efficacy. This effectiveness applies as well not only in environments with moderate numbers of VMs as in complex environments with different VM configurations and varying workload patterns. It is noteworthy that GWO-CSA is a perfect choice for cloud computing platforms where task demands and resources can change very significantly.GPO-ALBA exhibited remarkable capability, being quite close to GWO-CSA in many of the tasks, especially with less number of tasks. Nevertheless, its efficiency was lower compared to larger sets of tasks, and it seems that it has some limitations in handling a large volume of work.Different from PSO and ACO algorithms that are good enough for smaller sizes task scenarios but are degraded as the number of tasks and VMs increase. They face the challenges of keeping efficiency while handling high workloads, which show insufficient scalability or adaptability in upper cloud systems.To sum it up, this paper stresses on giving due attention while choosing suitable task scheduling algorithms in cloud computing. GWO-CSA achieved the status of an extremely efficient and adaptive algorithm which is suitable for diverse requirements of cloud computing. This offers the ability to regulate high with low when the conditions are diverse; and hence, becomes a valuable tool for improving the use of cloud computing resources yielding better overall performance.

REFERENCES

[1] Houssein, E. H., et al. "Task Scheduling in Cloud Computing based on Meta-heuristics: Review, Taxonomy, Open Challenges, and Future Trends." *Swarm and Evolutionary Computation*, vol. 62, Apr. 2021, p. 100841. *ScienceDirect*, doi:10.1016/j.swevo.2021.100841.

[2] Potluri, S., and K. S. Rao. "Optimization model for QoS based task scheduling in cloud computing environment." *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 18, no. 2, May 2020, pp. 1081-1088. *IEEE Xplore*, doi:10.11591/ijeecs.v18.i2.pp1081-1088.

[3] Singh, H., A. Bhasin, and P. Kaveri. "SECURE: Efficient resource scheduling by swarm in cloud computing." *Journal of Discrete Mathematical Sciences & Cryptography*, vol. 22, no. 2, Feb. 2019, pp. 127-137. *Taylor & Francis Online*, doi:10.1080/09720529.2019.1576334.

[4] Abed-alguni, B. H., and F. Alkhateeb. "Novel Selection Schemes for Cuckoo Search." *Arabian Journal for Science and Engineering*, vol. 42, no. 8, Aug. 2017, pp. 3635-3654. *SpringerLink*, doi:10.1007/s13369-017-2663-3.

[5] Zhou, G., W. Tian, and R. Buyya. "Deep Reinforcement Learning-based Methods for Resource Scheduling in Cloud Computing: A Review and Future Directions." *arXiv*, 9 May 2021. *arXiv.org*, http://arxiv.org/abs/2105.04086. Accessed 29 Jul. 2022.

[6] Wang, Y., et al. "Multi-Objective Workflow Scheduling with Deep-Q-Network-Based Multi-Agent Reinforcement Learning." *IEEE Access*, vol. 7, 2019, pp. 39974-39982. *IEEE Xplore*, doi:10.1109/ACCESS.2019.2902846.

[7] Bezdan, T., et al. "Multi-objective task scheduling in cloud computing environment by hybridized bat algorithm." *Journal of Intelligent & Fuzzy Systems*, vol. 42, no. 1, Dec. 2021, pp. 411-423. *IOS Press*, doi:10.3233/JIFS-219200.

[8] Jena, R. K. "Task scheduling in cloud environment: A multi-objective ABC framework." *Journal of Information and Optimization Sciences*, vol. 38, no. 1, Jan. 2017, pp. 1-19. *Taylor & Francis Online*, doi:10.1080/02522667.2016.1250460.

[9] Shalu, and D. Singh. "Artificial neural network-based virtual machine allocation in cloud computing." *Journal of Discrete Mathematical Sciences & Cryptography*, vol. 24, no. 6, Aug. 2021, pp. 1739-1750. *Taylor & Francis Online*, doi:10.1080/09720529.2021.1878626.

[10] Ahmad, O., and R. Z. Khan. "Pso-Based Task Scheduling Algorithm Using Adaptive Load Balancing Approach for Cloud Computing Environment." *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 8, no. 11, 2019, p. 7. *TheSaiOrg*, http://thesai.org/Publications/ViewPaper?Volume=8&Issue=11&Code=IJACSA&SerialNo=7.

[11] SKA, Manish Kumar Mukhija, and Pooja Singh "A Security Approach to Manage a Smart City's Image Data on Cloud," AI-Centric Smart City Ecosystems: Technologies, Design and Implementation (1st ed.), PP: 68-82, (2022). CRC Press. https://doi.org/10.1201/9781003252542.

[12] SKA and Abha Jadaun. "Design and Performance Assessment of Light Weight Data Security System for Secure Data Transmission in IoT", Journal of Network Security, 2021, Vol-9, Issue-1, PP: 29-41.

[13] Pratiksha Mishra and SKA. "Design & Performance Assessment of Energy Efficient Routing Protocol Using Improved LEACH", International Journal of Wireless Network Security, 2021, Vol-7, Issue-1, PP: 17-33.

[14] Rajput, B. S. .; Gangele, A. .; S. K. . Numerical Simulation and Assessment of Meta Heuristic Optimization Based Multi Objective Dynamic Job Shop Scheduling System. *ijfrcsce* 2022, *8*, 92-98.

[15] Khandelwal, R., Mukhija, M.K. and S.K., 2021. Numerical simulation and performance assessment of improved particle swarm optimization based request scheduling in edge computing for IOT applications. *New Arch-International Journal Of Contemporary Architecture*, *8*(2), pp.155-169.

[16] S.K. and Kumar, A., 2018. Implementation of new cryptographic encryption approach for trust as & service (TAAS) in cloud environment. *International Journal of Computers and Applications*, *4*(8), pp.2250-1797.

[17] Bhatt, V., Diwan, H.S., S.K. and Saini, Y., 2021. Empowering ML Work-Flow with DevOps within Micro Service Architecture and Deploying A Hybrid-Multi Cloud, Maintaining CI/CD Pipeline: An Open Shift Orchestration of ML-OPS. *New Arch-International Journal Of Contemporary Architecture*, *8*(2), pp.147-154.