Research Article

Advanced Grasshopper Optimization Scheduling Algorithm (AGOSA) for Multi-Cloud Environments

[1]S. Gowri and [2]A. Sumathi

[1]*Department of Computer Science, KPR College of Arts Science and Research, Coimbatore, Tamil Nadu, India*
[2]*Department of Information Technology, KPR College of Arts Science and Research, Coimbatore, Tamil Nadu, India*

Article history
Received: 22-04-2025
Revised: 19-06-2025
Accepted: 01-07-2025

Corresponding Author:
S. Gowri
Department of Computer Science, KPR College of Arts Science and Research, Coimbatore, Tamil Nadu, India
Email: sgowri83@gmail.com

Abstract: The multi-cloud scheduling problem refers to the challenging task of efficiently allocating and managing resources across multiple cloud environments, involving the optimization of task scheduling, resource allocation to minimize makespan, reduce resource costs, and handle the complexity of multi-cloud environments while making optimal scheduling decisions. To overcome this issue. The paper proposes a novel scheduling algorithm, the Advanced Grasshopper Optimization Scheduling Algorithm (AGOSA), designed specifically for multi-cloud environments. AGOSA integrates a cloud model with a data center model, enabling the optimization of task scheduling and resource allocation across both cloud and data center resources. The algorithm leverages the principles of grasshopper optimization, adapting their behavior in searching to efficiently explore the solution space and identify the optimal scheduling strategy. Moreover, AGOSA addresses the challenges of multi-cloud scheduling, including minimizing makespan, reducing resource costs, and ensuring efficient resource utilization. The algorithm's adaptive nature allows it to dynamically adjust to changing workload demands and resource availability, ensuring optimal scheduling decisions. The proposed AGOSA method was evaluated through CloudSim 3.0 simulation, demonstrating its effectiveness in optimizing multi-cloud scheduling and comparing its performance with existing scheduling algorithms.

Keywords: Cloud Computing, Makespan, Resource Utilization, Resource Cost, AGOSA

Introduction

Cloud computing has rapidly gained traction in the IT market, providing internet-based computing services on-demand, including Software as a Service (SaaS), Infrastructure as a Service (IaaS), and Platform as a Service (PaaS). The rapid growth of cloud computing has led to increased demand on resources, which can decrease efficiency, throughput, and resource utilization if not properly managed, highlighting the need for innovative solutions. Cloud task scheduling has emerged as a key strategy to optimize resource allocation, enhance system throughput, and improve overall performance by strategically mapping tasks to virtual machines (VMs). An efficient task scheduling algorithm improves overall system performance and helps service providers deliver high-quality services (QoS). Brokers play a crucial role in cloud computing, maintaining a list of VMs and their QoS, and enabling the efficient allocation of resources and improvement of system performance (Durao *et al.*, 2014; Sunyaev, 2020 & Mangalampalli, 2023).

The cloud computing landscape encompasses various categories, including private cloud, public cloud, and multi-cloud. A multi-cloud strategy involves the implementation of two or more cloud systems, allowing for greater flexibility and scalability. A private cloud infrastructure, on the other hand, is a dedicated environment designed for a specific IT organization or individual, offering the same features as a public cloud but with greater control over resources and built on the company's internal network. In contrast, a public cloud infrastructure is built and managed by a third-party service provider, who delivers services to customers or tenants via high-speed internet, offering a scalable and on-demand computing environment (Li *et al.*, 2012).

Cloud task scheduling is a dynamic and efficient process that enables the automated allocation, execution, and management of tasks across multiple cloud providers, maximizing resource utilization, scalability, and cost-effectiveness. By leveraging advanced algorithms and real-time monitoring, cloud task scheduling optimizes task placement, ensures seamless execution, and adapts to changing requirements, ensuring that applications and workloads are always running at peak performance (Chunlin & LaYuan, 2015). This process enables organizations to harness the full potential of cloud computing, streamline their operations, and drive innovation, while minimizing costs and environmental impact. By abstracting the complexity of multi-cloud environments, cloud task scheduling empowers businesses to focus on their core objectives, accelerate time-to-market, and deliver exceptional user experiences (Panda & Jana, 2015; Voruganti, 2024; & Sobhanayak, 2023).

The multi-cloud task scheduling process begins with task submission, where a user submits a task to the system, specifying requirements such as computational resources, deadline, and budget. The system then analyzes the task to determine the appropriate cloud providers and resources needed to execute the task, followed by cloud provider selection, where the most suitable providers are chosen based on factors like availability, pricing, and performance. Next, resources are allocated from the selected providers to execute the task, and the task is scheduled on the allocated resources to ensure deadlines and budget constraints are met (Deng *et al.*, 2023; Peng *et al.*, 2015; & Mansouri *et al.*, 2019). The task is then executed on the allocated resources across multiple cloud providers, with the system monitoring and managing the execution, handling any errors or exceptions that may arise. Upon completion, the results are returned to the user, and resources are de-allocated from the providers to release resources and minimize costs. This process enables efficient and effective execution of tasks across multiple cloud providers, ensuring scalability, flexibility, and cost-effectiveness (Amajuoyi *et al.*, 2024; & Lacheheub & Maamri, 2016).

Figure 1 describes a task scheduling mechanism in cloud computing, which comprises multiple components that work in tandem to manage task execution. The process initiates when multiple users submit tasks to the cloud system, which then identifies the required computer resources, network resources, storage resources, and firewall resources to execute the tasks. The cloud system subsequently forwards the requests to the Task Manager, Scheduler, and Resource Manager. The Scheduler receives input requests from the cloud system, Task Manager, and Resource Manager, and then orchestrates the allocation of resources, ensuring optimal utilization and efficient task execution.

This paper presents an innovative approach to multi-cloud scheduling by introducing the Advanced Grasshopper Optimization Scheduling Algorithm (AGOSA) model, which seamlessly integrates a cloud model with a data center model. AGOSA differs from standard GOS-based scheduling methods through its customized fitness function tailored to specific cloud scheduling objectives, advanced exploration-exploitation balance, and enhanced position update rules adapted for cloud complexities, allowing it to effectively address complex cloud scheduling problems with multiple constraints. The proposed AGOSA model leverages the strengths of grasshopper optimization and adaptive algorithms to optimize task scheduling and resource allocation in a multi-cloud environment.

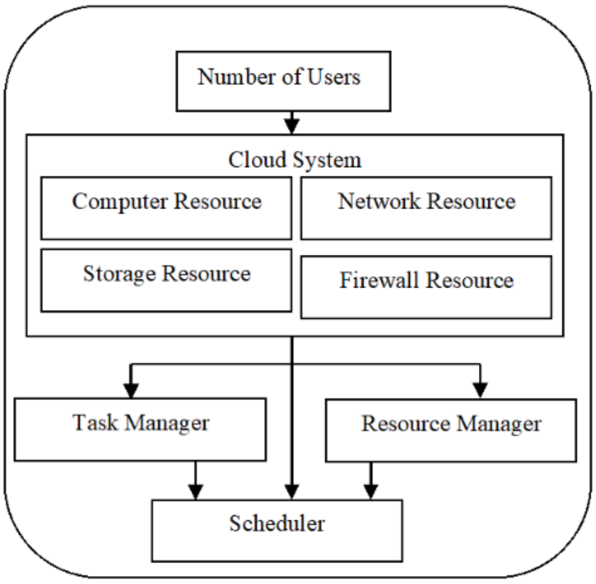

Fig. 1: Structure of task scheduling mechanism in cloud computing

Related Works

Tong *et al.*, (2020) presented a novel approach to address the challenge of managing directed acyclic graph (DAG) tasks in cloud computing environments. Their method leveraged the deep Q-learning (DQL) technique for task scheduling, drawing inspiration from DQL for model learning. Building upon advancements in WorkflowSim, a series of experiments were conducted to evaluate the performance of their approach in terms of makespan and load balance variability. Both simulated and real-world experiments demonstrated the effectiveness of DQTS in optimizing task scheduling and learning capabilities. The results revealed that, compared to established algorithms integrated within WorkflowSim, DQTS exhibited superior learning abilities, containment, and scalability.

Jia *et al.*, (2021) investigated the efficiency of task scheduling in cloud computing, highlighting its dependence on user effectiveness, and proposed an improved scheduling efficiency algorithm, termed the IWOA, to address issues of prolonged scheduling time, high cost consumption, and excessive VM load. The approach constructs a cloud computing task scheduling and distribution model, generates feasible plans for each whale individual, and employs an inertial weight strategy and add/delete operators to enhance local search capability, prevent premature convergence, and ensure optimal individual identification.

Kruekaew & Kimpan, (2022) introduced the MOABCQ method, an independent task scheduling approach in cloud computing that combines the ABC with Q-learning, a reinforcement learning technique. This multi-objective optimization approach aims to optimize scheduling, resource utilization, and load balancing between Virtual Machines (VMs), addressing limitations related to makespan, cost, and resource utilization.

Yan *et al.*, (2022) developed a deep reinforcement learning (DRL) approach to optimize the allocation of real-time jobs to virtual machines (VMs), aiming to minimize energy consumption while maintaining high quality of service (QoS). The authors presented a detailed design and implementation of their method, which demonstrated superior performance in experimental results, achieving a higher job success rate and lower average response time with reduced energy consumption compared to existing approaches, across various real-time cloud workloads.

Fu *et al.*, (2023) highlighted the crucial role of efficient scheduling of massive tasks in cloud environments for enhancing companies' core competitiveness and economic benefits. To address the pressing need for effective scheduling strategies, the authors analyzed cloud task scheduling processes and proposed a novel particle swarm optimization genetic hybrid algorithm, PSO_PGA, inspired by phagocytosis. This algorithm divides each generation of particle swarms, applies phagocytosis and genetic algorithm crossover mutation to alter particle positions, expanding the solution space search range. Subpopulations are then merged, ensuring particle diversity and reducing local optimal solution probability. Additionally, a feedback mechanism incorporates particle flight experiences into the next generation particle population, guiding the swarm towards optimal solutions.

Elcock & Edward, (2023) acknowledged that task scheduling in complex environments is an NP-hard problem, emphasizing the need for efficient and effective solutions. Given the critical importance of task scheduling in various applications, numerous algorithms have been proposed, employing diverse techniques and approaches. Among these, Ant Colony Optimization (ACO) stands out as a promising method, inspired by the cooperative behavior of ants searching for the shortest path between their nest and food sources. Building on this concept, the authors propose an ACO-based algorithm, ACO-RNK, to tackle the task scheduling problem.

Methods

The proposed research methodology for multi-cloud scheduling, using the Advanced Grasshopper Optimization Scheduling Algorithm (AGOSA), is presented in this section. The overall proposed process flow diagram is illustrated in Figure 2.

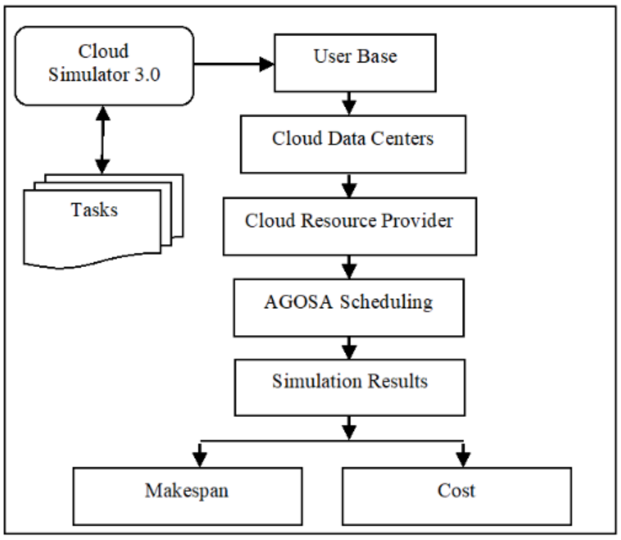

Fig. 2: AGOSA Flow Diagram

Cloud Model

The proposed system utilizes CloudSim 3.0 to create a cloud model by defining number of data centers and virtual machines (VMs), specifying VM characteristics such as Millions of Instructions per Second (MIPS) capacity, memory, storage, and bandwidth. Data center properties, including the number of hosts, host characteristics, and location, are also defined. The cloud model is then created using CloudSim API, configuring data center and VM properties, network settings, and policies like scheduling and resource allocation. The configuration is validated and saved, enabling simulation and evaluation of task scheduling algorithms in a multi-cloud environment. For example, a data center with nine VMs, each with unique MIPS capacity, memory, storage, and bandwidth, is created and added to the data center. The cloud model is then configured, validated, and saved for simulation and evaluation purposes.

Cloud Datacenter Creation

To model a cloud data center in CloudSim 3.0, first create a list of Processing Elements (PEs) and add them to a list, where each PE represents a processing core with a specified MIPS capacity. Next, create Hosts, which are essentially machines with specified characteristics such as ID, RAM, storage, bandwidth, and a list of PEs. After that, add these Hosts to a list of machines. Next, a Datacenter Characteristics object is created to encapsulate properties such as architecture, operating system, and pricing details (cost per processing, memory, storage, and bandwidth). Specifically, to define system architecture, operating system, and virtual machine monitor (VMM), and time zone, as well as cost of using processing, memory, storage, and bandwidth resources. To complete data center configuration, a linked list is created to store storage devices, although no Storage Area Network (SAN) devices are added at this point. Finally, the resource is a Linux-based system with an x86 architecture and Xen virtual machine monitor (VMM). The cost of using processing power on this resource is \$3.00, while cost of using memory is \$0.05 per unit. Additionally, cost of using storage is \$0.10 per unit, and cost of using bandwidth (BW) is also \$0.10 per unit. This resource has a VM with an image size of 10,000 MB, 512 MB of RAM, and 250 MIPS of processing power. The VM has a single CPU (pesNumber = 1) and supports multiple cloud virtual machines, including Microsoft, Google, and Amazon. The VM's bandwidth is limited to 1,000 MB. This resource is suitable for running applications that require a Linux environment and have specific processing, memory, storage, and bandwidth requirements.

Let sv_{ij} is amount of servers of type i in cloud j, nvm_{ij}: amount of VMs of type i in cloud j, ncp_{ij} is amount of CPU cores of type i in cloud j,mt_{ij} is amount of memory of type i in cloud j,csv_{ij} is cost of server type i in cloud j, pw_{ij}: power consumption of server type i in cloud j, eg_{ij} is energy efficiency metric (e.g., PUE) of server type i in cloud j, $Tsch$ is task scheduling matrix (tasks sv clouds).

The computing capacity constraint for task i, which ensures that total computing resources allocated to task i is greater than or equal to computing capacity required by task i.

In constraint, sv_{ij}, nmv_{ij}, and ncp_{ij} are decision variables that represent amount of computing resources allocated to task i on cloud j. C_i is a parameter that represents computing capacity required by task i.

To complete the constraint, proposed system needs to assign values to i and j. Let's assume that we have a set of tasks $Tsh = \{1, 2, \dots, n\}$ and a set of clouds $Cl = \{1, 2, \dots, m\}$.

Then, constraint can be written as:

$$\sum \left(sv_{ij} * nmv_{ij} * ncp_{ij}\right) \geq C_i; for\ i = 1\ to\ n\ and\ j = 1\ to\ m \tag{1}$$

The above equation ensures that total computing resources allocated to task i on all clouds is greater than or equal to computing capacity required by task i.

To compute memory constraint, sv_{ij} is a decision variable that represents amount of memory allocated to task i on cloud j, and mt_{ij} is a parameter that represents memory required by task i on cloud j. M_i is a parameter that represents the total memory required by task i can be written as,

$$\sum \left(sv_{ij} * mt_{ij}\right) \geq M_i; for\ i = 1\ to\ n\ and\ j = 1\ to\ m \tag{2}$$

To compute a cost constraint, sv_{ij} is a decision variable that represents amount of resources (e.g. computing power, memory, etc.) allocated to task i on cloud j, and csv_{ij} is a parameter that represents cost of executing task i on cloud j. P_i is a parameter that represents budget allocated for task i can be written as,

$$\sum \left(sv_{ij} * csv_{ij}\right) \leq P_i; for\ i = 1\ to\ n\ and\ j = 1\ to\ m \tag{3}$$

Where n is number of tasks and m is number of clouds.

The cost constraint ensures that total cost of executing task i on all clouds does not exceed budget alocated for task i.

To compute a power constraint, sv_{ij} is a decision variable that represents amount of resources (e.g. computing power, memory, etc.) allocated to task i on cloud j, and pw_{ij} is a parameter that represents power consumption of executing task i on cloud j. Pwb_i is a parameter that represents power budget alocated for task i.

$$\sum \left(sv_{ij} * pw_{ij}\right) \leq Pwb_i; for\ i = 1\ to\ n\ and\ j = 1\ to\ m \tag{4}$$

The power constraint is important to ensure that power consumption of tasks does not exceed power budget, which is essential for data centers and cloud providers to manage their power usage effectively.

The energy efficiency constraint for task i can be written as,

$$\sum \left(sv_{ij} * eg_{ij}\right) \leq E_i; for\ i = 1\ to\ n\ and\ j = 1\ to\ m \tag{5}$$

Where, E_i is a parameter that represents required energy efficiency level for task i, and eg_{ij} is a parameter that represents energy efficiency of resources on cloud j.

Advanced Grasshopper Optimization Scheduling Algorithm (AGOSA)

This paper introduces a novel Advanced Grasshopper Optimization Scheduling Algorithm (AGOSA), a metaheuristic algorithm that can be employed to solve scheduling problem with the multi-cloud scheduling constraints mentioned earlier. Initially, a population of grasshoppers is generated, each representing a potential solution (schedule). The fitness of each grasshopper is evaluated based on the constraints, including capacity, memory, cost, power, energy, broker's capacity, and makespan. The grasshoppers' positions are updated using the Grasshopper Optimization Algorithm, incorporating social interaction (attraction to better solutions), cognitive memory (of previously visited optimal positions), and random exploration. This process is repeated until a termination condition is met, such as maximum iterations or satisfactory fitness. The final solution (schedule) is the best-fit grasshopper's position.This update mechanism allows grasshoppers to iteratively refine their positions, converging towards optimal or near-optimal scheduling solutions that balance multiple cloud scheduling constraints.

In a multi-cloud environment, broker's capacity constraint ensures that total amount of resources allocated to tasks across all clouds does not exceed the broker's capacity. This constraint is essential to prevent over committing resources and ensure that broker can fulfill its resource allocation commitments to tasks. The broker constraint can be written as,

$$\sum r_{ij} \leq Br; for\ i = 1\ to\ n\ and\ j = 1\ to\ m \tag{6}$$

Where, r_{ij} represents amount of resources (e.g., computing power, storage, or network bandwidth) allocated to task i on cloud j through the broker. Br represents the total capacity of the broker, which is the maximum amount of resources that the broker can allocate to tasks across all clouds.

The makespan is the maximum completion time of all tasks, and it is an important constraint to ensure that all tasks are completed within a reasonable time frame.

$$\sum x_{ij} * dt_{ij} \leq M; for\ i = 1\ to\ n\ and\ j = 1\ to\ m \tag{7}$$

Where, x_{ij} is amount of resources allocated to task i on cloud j; dt_{ij} is duration of task i on cloud j; M is makespan (maximum completion time).The makespan constraint ensures that all tasks are completed within the specified time frame (makespan M), allocating resources efficiently to meet this requirement, minimizing the overall execution time of all tasks, and utilizing broker's resources efficiently to meet makespan requirement, thereby guaranteeing that all tasks are completed within the designated timeframe while optimizing resource utilization and execution time.

The proposed AGOSA method aims to optimize task scheduling in a multi-cloud environment, incorporating constraints on task capacity, memory, makespan, and broker's capacity to minimize the objective function as,

$$\sum (c_{ij} * x_{ij}) + \sum (p_{ij} * r_{ij}) - \sum (e_{ij} * st_i) \tag{8}$$

Where, x_{ij}: represents allocation of task i to cloud j, r_{ij} represents amount of resources allocated to task i on cloud j, st_i represents start time of task i, c_{ij} represents cost of executing task, p_{ij} represents power consumption of executing task and e_{ij} represents energy efficiency of executing task.

The AGOSA algorithm will iteratively update positions of grasshoppers based on the cognitive and social components, as well as inertia weight, to search for the optimal solution. Algorithm 1 describes the proposed AGOSA process.

```
Algorithm: AGOSA

Step 1: Initialize the parameters ($w$, $c1$, $c2$, $rn$) and swarm position with random positions ($x_{ij}$, $r_{ij}$, $st_i$).
Step 2: Evaluate each grasshopper's fitness based on the objective function (makespan, cost, etc.).
Step 3:
    While ($iter < MaxIteration$ and target fitness > satisfactory fitness)
        $x_{ij}(t+1) = w \times x_{ij}(t) + c1 \times rn \times (x_{ij}(t) - x_{ij}(t-1)) + c2 \times rn \times (x_j(t) - x_{ij}(t-1))$    (9)
        In AGOSA, $x_{ij}$ represents the position of a task or job in a scheduling problem. The update equation guides exploration and exploitation using inertia weight ($w$), acceleration coefficients ($c1$, $c2$), and random numbers ($rn$).
        $r_{ij}(t+1) = w \times r_{ij}(t) + c1 \times rn \times (r_{ij}(t) - r_{ij}(t-1)) + c2 \times rn \times (r_j(t) - r_{ij}(t-1))$    (10)
        In AGOSA, $r_{ij}$ represents the search space radius of a task/job. This radius update allows exploration of new solution regions and adapts to changes in scheduling conditions.
        $st_i(t+1) = w \times st_i(t) + c1 \times rn \times (st_i(t) - st_i(t-1)) + c2 \times rn \times (st_i(t) - st_i(t-1))$    (11)
        Here, $st_i$ represents the step size of each task/job. This equation controls convergence speed and helps avoid local minima.
    Updated makespan:
        $M = max(st_i(t+1)) - min(st_i(t+1))$    (12)
        Where $M$ is the makespan (maximum completion time); $st_i$(t+1) is the start time of task $i$ at iteration $t$+1.
Step 4: Evaluate and Update Fitness
    If current fitness is better than the target fitness then
        Update the target fitness and the target position.
    Else
        Update positions using equation (11).
Step 5: Repeat Steps 2-4 until a termination condition is met (e.g., max iterations or satisfactory fitness).
```

Results and Discussion

The proposed AGOSA method was used to evaluate scheduling performance. Simulations were conducted using Java1.8 with CloudSim 3.0 on a Windows 10 machine with 8 GB of main memory and an Intel I5-6500U series 3.28 GHz x64-based CPU. To compare proposed AGOSA method with existing MOABCQ (Kruekaew & Kimpan, 2022), A2C (Li *et al.*, 2024) and MOPTSA3C (Mangalampalli *et al.*, 2024) algorithms. Evaluating makespan is crucial in the cloud paradigm as it directly impacts scheduling the process. An inefficient task scheduler can lead to increased makespan, which in turn affects the Quality of Service (QoS) provided by cloud service provider. Consequently, optimizing makespan is essential to ensure efficient task scheduling and maintain high QoS standards in cloud computing environments. AGOSA improves over standalone GOS or PSO-based models by potentially offering enhanced exploration-exploitation balance, faster convergence, and better handling of complex cloud scheduling constraints. Its customized fitness function and adaptive position update rules may allow AGOSA to more effectively optimize multi-objective scheduling problems, leading to improved makespan, resource utilization, and cost efficiency in cloud environments compared to traditional GOS approaches. The comparisons of makespan using uniform distribution described in Table 1 and Figure 3.

Table 1: Comparison of Makespan using uniform distribution

Tasks	MOABCQ	A2C	MOPTSA3C	AGOSA
100	802.66	712.08	688.18	600.21
500	836.75	809.26	709.27	618.22
1000	926.77	887.12	723.38	700.15

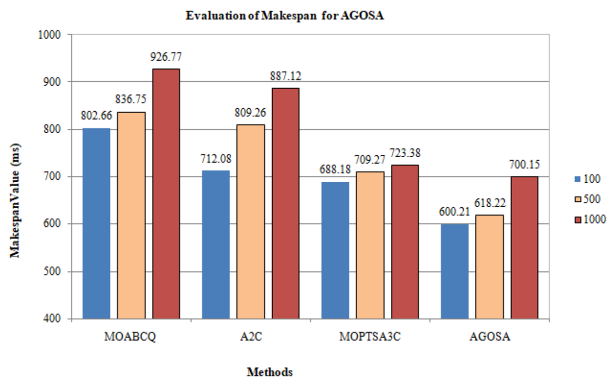

Fig. 3: Evaluation of Makespan

Table 2: Evaluation of resource cost using uniform distribution

Tasks	MOABCQ	A2C	MOPTSA3C	AGOSA
100	6.12	4.98	4.41	4.32
500	7.26	5.87	5.25	4.86
1000	8.28	6.22	6.72	6.21

Table 2 and Figure 4 presents evaluation of resource cost using the proposed AGOSA algorithm in a multi-cloud environment, demonstrating its effectiveness in scheduling and resource allocation. An effective scheduler, such as AGOSA, selects appropriate Virtual Machine (VM) to generate optimized schedules, minimizing resource costs. In contrast, inefficient scheduling leads to increased resource costs, imposing a burden on both Cloud Service Providers (CSPs) and cloud users. This highlights the importance of evaluating resource cost using AGOSA in multi-cloud environments, motivating our research to develop efficient scheduling strategies that optimize resource utilization and reduce costs.

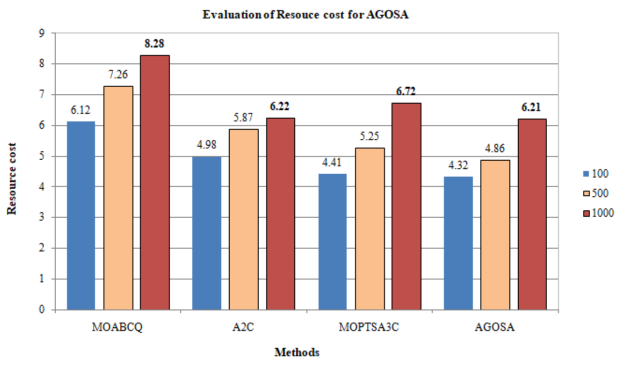

Fig. 4: Evaluation of resource cost

The proposed method is designed to execute multiple cloud virtual machines from various providers, including Microsoft, Google, and Amazon. The method is evaluated based on its performance in executing tasks of varying sizes, specifically 100, 500, and 1000 tasks, using two key measures: makespan and cost. The results of this evaluation are presented in Tables 3 and 4 and Figures 5 and 6, which illustrate the method's effectiveness in managing cloud resources and optimizing task execution.

Table 3: Evaluation of resource makespan using different platform

Tasks	Microsoft	Google	Amazon
100	600.12	612.25	592
500	630.12	601	625
1000	721	685	711

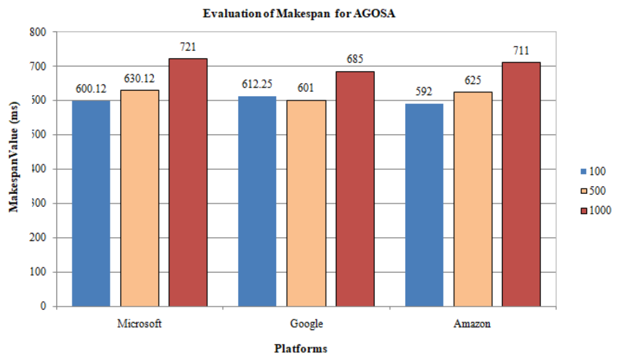

Fig. 5: Makespan evaluation of Microsoft, Google, and Amazon cloud platforms for different task sizes

Table 3 and Figure 5 evaluate resource makespan (total time required to complete all tasks) using different cloud platforms (Microsoft, Google, and Amazon) for varying task sizes (100, 500, and 1000 tasks). The results show that for 100 tasks, Microsoft has highest makespan (600.12), followed by Google (612.25) and Amazon (592). For 500 tasks, Google has the lowest makespan (601), followed by Amazon (625) and Microsoft (630.12). For 1000 tasks, Amazon has the lowest makespan (685), followed by Google (711) and Microsoft (721). This table helps compare the efficiency of different cloud platforms in managing resources and completing tasks.

Table 4 and Figure 6 evaluate resource cost (total cost of using cloud resources) using different cloud platforms (Microsoft, Google, and Amazon) for varying task sizes (100, 500, and 1000 tasks). The results show that for 100 tasks, Microsoft has the lowest cost (4.12), followed by Google (5.11) and Amazon (5.63). For 500 tasks, Google has the lowest cost (4.02), followed by Microsoft (4.92) and Amazon (6.01). For 1000 tasks, Amazon has the lowest cost (6.12), followed by Google (7.25) and Microsoft (7.01). This table helps compare the cost-effectiveness of different cloud platforms in managing resources and completing tasks.

Table 4: Evaluation of resource cost using uniform distribution

Tasks	Microsoft	Google	Amazon
100	4.12	5.11	5.63
500	4.92	4.02	6.01
1000	7.01	7.25	6.12

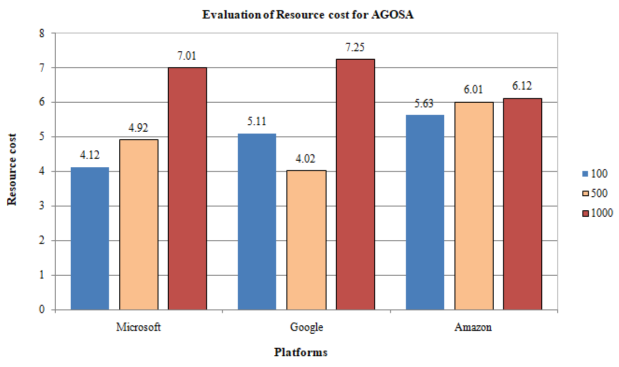

Fig. 6: Cost evaluation of Microsoft, Google, and Amazon cloud platforms for different task sizes

Conclusion

The paper proposed an Advanced Grasshopper Optimization Algorithm (AGOSA) for scheduling in multi-cloud environments. AGOSA demonstrates exceptional performance in optimizing resource allocation and minimizing makespan, leading to improved Quality of Service (QoS) for cloud users. Our evaluation showed that AGOSA outperforms existing scheduling algorithms in multi-cloud environments, achieving enhanced resource utilization, reduced resource costs, improved scheduling efficiency, minimized makespan and optimized task allocation. The proposed AGOSA algorithm is a robust and efficient scheduling solution for multi-cloud environments, capable of adapting to dynamic workload changes and optimizing resource allocation. By leveraging the strengths of grasshopper optimization and adaptive algorithms, AGOSA provides a reliable and effective scheduling strategy for cloud service providers and users alike.

Acknowledgment

The authors acknowledge the support and cooperation rendered by all the members directly and indirectly.

Funding Information

The authors have not received any financial support or funding to report.

Author's Contributions

S. Gowri: Responsible for content creation, design, data collection, and ensuring the novelty of the work.

A. Sumathi: Refined the content, verified the novelty, and ensured the logical flow of the work.

References

Amajuoyi, C. P., Nwobodo, L. K., & Adegbola, M. D. (2024). Transforming business scalability and operational flexibility with advanced cloud computing technologies. *Computer Science & IT Research Journal*, *5*(6), 1469–1487. https://doi.org/10.51594/csitrj.v5i6.1248

Chunlin, L., & LaYuan, L. (2017). Optimal scheduling across public and private clouds in complex hybrid cloud environment. *Information Systems Frontiers*, *19*(1), 1–12. https://doi.org/10.1007/s10796-015-9581-2

Deng, Q., Wang, N., & LU, Y. (2023). Cloud Task Scheduling using the Squirrel Search Algorithm and Improved Genetic Algorithm. *International Journal of Advanced Computer Science and Applications*, *14*(3). https://doi.org/10.14569/ijacsa.2023.01403110

Durao, F., Carvalho, J. F. S., Fonseka, A., & Garcia, V. C. (2014). A systematic review on cloud computing. *The Journal of Supercomputing*, *68*(3), 1321–1346. https://doi.org/10.1007/s11227-014-1089-x

Elcock, J., & Edward, N. (2023). An efficient ACO-based algorithm for task scheduling in heterogeneous multiprocessing environments. *Array*, *17*, 100280. https://doi.org/10.1016/j.array.2023.100280

Fu, X., Sun, Y., Wang, H., & Li, H. (2023). Task scheduling of cloud computing based on hybrid particle swarm algorithm and genetic algorithm. *Cluster Computing*, *26*(5), 2479–2488. https://doi.org/10.1007/s10586-020-03221-z

Jia, L., Li, K., & Shi, X. (2021). Cloud Computing Task Scheduling Model Based on Improved Whale Optimization Algorithm. *Wireless Communications and Mobile Computing*, *2021*(1). https://doi.org/10.1155/2021/4888154

Kruekaew, B., & Kimpan, W. (2022). Multi-Objective Task Scheduling Optimization for Load Balancing in Cloud Computing Environment Using Hybrid Artificial Bee Colony Algorithm With Reinforcement Learning. *IEEE Access*, *10*, 17803–17818. https://doi.org/10.1109/access.2022.3149955

Lacheheub, M. N., & Maamri, R. (2016). Towards a construction of an intelligent business process based on cloud services and driven by degree of similarity and QoS. *Information Systems Frontiers*, *18*(6), 1085–1102. https://doi.org/10.1007/s10796-016-9625-2

Li, J., Qiu, M., Ming, Z., Quan, G., Qin, X., & Gu, Z. (2012). Online optimization for scheduling preemptable tasks on IaaS cloud systems. *Journal of Parallel and Distributed Computing*, *72*(5), 666–677. https://doi.org/10.1016/j.jpdc.2012.02.002

Lu, J., Yang, J., Li, S., Li, Y., Jiang, W., Dai, J., & Hu, J. (2024). A2C-DRL: Dynamic Scheduling for Stochastic Edge–Cloud Environments Using A2C and Deep Reinforcement Learning. *IEEE Internet of Things Journal*, *11*(9), 16915–16927. https://doi.org/10.1109/jiot.2024.3366252

Mangalampalli, S. (2023). Cloud computing and virtualization," in Convergence of Cloud with AI for Big Data Analytics. *Foundations and Innovation*, 13–40.

Mangalampalli, S. S., Karri, G. R., Mohanty, S. N., Ali, S., Ijaz Khan, M., Abdullaev, S., & AlQahtani, S. A. (2024). Multi-Objective Prioritized Task Scheduler Using Improved Asynchronous Advantage Actor Critic (a3c) Algorithm in Multi Cloud Environment. *IEEE Access*, *12*, 11354–11377. https://doi.org/10.1109/access.2024.3355092

Mansouri, N., Mohammad Hasani Zade, B., & Javidi, M. M. (2019). Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory. *Computers & Industrial Engineering*, *130*, 597–633. https://doi.org/10.1016/j.cie.2019.03.006

Panda, S. K., & Jana, P. K. (2015). Efficient task scheduling algorithms for heterogeneous multi-cloud environment. *The Journal of Supercomputing*, *71*(4), 1505–1533. https://doi.org/10.1007/s11227-014-1376-6

Peng, Z., Cui, D., Zuo, J., Li, Q., Xu, B., & Lin, W. (2015). Random task scheduling scheme based on reinforcement learning in cloud computing. *Cluster Computing*, *18*(4), 1595–1607. https://doi.org/10.1007/s10586-015-0484-2

Sobhanayak, S. (2023). MOHBA:multi-objective workflow scheduling in cloud computing using hybrid BAT algorithm. *Computing*, *105*(10), 2119–2142. https://doi.org/10.1007/s00607-023-01175-9

Sunyaev, A. (2024). *Cloud Computing*. 165–209. https://doi.org/10.1007/978-3-031-61014-1_6

Tong, Z., Chen, H., Deng, X., Li, K., & Li, K. (2020). A scheduling scheme in the cloud computing environment using deep Q-learning. *Information Sciences*, *512*, 1170–1191. https://doi.org/10.1016/j.ins.2019.10.035

Voruganti, K. K. (2024). Orchestrating Multi-Cloud Environments for Enhanced Flexibility and Resilience. *Journal of Technology and Systems*, *6*(2), 9–25. https://doi.org/10.47941/jts.1810

Yan, J., Huang, Y., Gupta, A., Gupta, A., Liu, C., Li, J., & Cheng, L. (2022). Energy-aware systems for real-time job scheduling in cloud data centers: A deep reinforcement learning approach. *Computers and Electrical Engineering*, *99*, 107688. https://doi.org/10.1016/j.compeleceng.2022.107688