MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1s, 2026, Pages: 409-416

Adaptive Graph Attention Reinforcement Scheduling with Hybrid Optimization for SLA-Aware Multi-Cloud Task Scheduling

M.Rupasri[1], G.P.S. Varma[2], Indukuri Hemalatha[3]
[1] Research Scholar, Dept. of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, A.P, India.
[2] Professor, Dept. of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, A.P, India.
[3] Professor, Dept. of Information Technology, S.R.K.R Engineering College, A.P, India.
e-mail: rupasriklu@gmail.com.

Abstract: Task scheduling is one of the most important considerations in a multi-cloud environment, given heterogeneous resources, dynamic workloads, and Service Level Agreement (SLA) requirements. Traditional heuristic and metaheuristic scheduling algorithms cannot achieve optimal performance in large-scale cloud infrastructures because they are not adaptable to changing system conditions. To overcome these constraints, this paper proposes an Adaptive Graph Attention Reinforcement Scheduling (AGARS) system for SLA-conscious multi-cloud task scheduling. The presented framework combines workload prediction using previous temporal transformers, embedding learning via a graph attention network (GAT), and reinforcement learning with the Soft Actor-Critic (SAC) algorithm. Moreover, an Augmented Lagrangian optimization module is also added to meet the required constraints, and a hybrid Harris Hawks Optimization-Differential Evolution (HHO-DE) strategy is used to improve the global search and solution optimization under heavy loads. It compared the proposed AGARS framework to baseline algorithms such as Round Robin, Genetic Algorithm, Particle Swarm Optimization, Deep Reinforcement Learning and GNN-based scheduling. Through experimental findings, AGARS achieves substantial performance improvements, including about a 40% reduction in execution time, a 65% reduction in SLA violations, a 30% increase in throughput, and a 27% decrease in energy usage compared to its counterparts. This set of results an effective and scalable tool for managing intelligent resources in a contemporary multi-cloud computing setup.

Keywords: Multi-Cloud Task Scheduling, Graph Attention Networks (GAT), Deep Reinforcement Learning, Service Level Agreement (SLA) Optimization, Hybrid Metaheuristic Optimization.

1. Introduction:

Cloud computing is now a technology used as the basis for the provision of computing resources that are scalable, such as storage, processing power, and networking resources, over the Internet. It also allows organizations to access computing resources when they need them without having to maintain costly infrastructures. In recent years, there has been growing interest in multi-cloud computing, in which organizations access the resources of several cloud providers simultaneously. Multi-clouds enhance reliability, availability, and flexibility, as well as minimizing vendor lock-in. Nevertheless, integrating heterogeneous cloud infrastructures presents several challenges, especially in task scheduling, resource allocation, and management of the service-level agreement (SLA) [1, 2]. The scheduling of tasks in the cloud context involves assigning incoming tasks to virtual machines (VMs) and satisfying various performance goals, including reducing the execution time, maximizing throughput, utilizing resources, and ensuring compliance with the service-level agreement (SLA). The Round Robin and First Come First Serve are traditional scheduling algorithms which are easy to compute and simple to understand but do not meet the needs of dynamic workload conditions and heterogeneous resource environments. These classical methods are no longer adequate because cloud infrastructures have become sufficiently large and complex to support dynamic and large-scale task scheduling issues [3]. Researchers have suggested numerous metaheuristic optimization methods for cloud scheduling to overcome these limitations. Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Firefly Optimization are some of the widely used algorithms that have been applied to enhance the performance of the scheduling process through exploration of high solution space and the identification of near-optimal assignment of tasks to resources. Although these approaches offer better optimization features than heuristic algorithms, they are usually characterized by slow convergence and reduced flexibility to fast-evolving workloads in real-time cloud setups [4], [5].

Owing to the rapid development of artificial intelligence technologies, machine learning-based scheduling models have become a potential solution to cloud resource management. Reinforcement learning (RL) methods allow learning systems to adapt the best scheduling policies by continuously interacting with the environment. RL-based schedulers are more likely to be applicable in complex cloud infrastructures because schedulers can adapt dynamically to the changing environments of the system. Recent research has demonstrated that reinforcement learning techniques are more efficient in scheduling performance regarding resource allocation and performance efficiency than traditional algorithms [6]. In particular, deep reinforcement learning (DRL) has received significant attention owing to its capacity to deal with highly dimensional state spaces and intricate decision-making processes. DRL models are capable of adapting scheduling policies that can simultaneously optimize various goals, such as execution time, cost, and energy usage. Other studies have shown that DRA-style scheduling systems perform better than conventional algorithms in large-scale distributed computing systems [7]. Although these are the benefits, DRL-based scheduling systems have several challenges. A key weakness is that it is difficult to model the multifaceted interactions between tasks and computing resources. Cloud environments have complex task, virtual machine, and network state dependencies. Traditional DRA models may not be able to effectively model such relationships, which decreases the efficiency of scheduling and incurs other resource allocation decisions that are suboptimal [8]. To address these problems, scholars have recently considered applying graph neural networks (GNNs) to reinforcement learning models. GNNs are structured to work with graph-based representations and can discover connections between entities that are in contact with each other. In the context of cloud computing, it is possible to model tasks and virtual machines as nodes in a graph, and the interactions between them as edges. GNNs can learn node embeddings through message-passing algorithms that learn the complex relationship between tasks and resources and enhance scheduling choices [9]. Energy efficiency is another factor that must be considered in contemporary cloud infrastructures. Large data centres are also significant electricity consumers, which increases the cost of operations and the environmental effects of big data systems. Cloud computing has made energy-aware scheduling algorithms a significant research topic. Achieving sustainable cloud computing systems requires intelligent scheduling structures to optimize performance and be energy-efficient [10]. Although current research has contributed to the field of cloud scheduling, available methods tend to concentrate on machine learning models or optimization algorithms alone. Hybrid intelligent scheduling frameworks that integrate decision models based on learning and sophisticated optimization algorithms are required to attain robust and scalable multi-cloud scheduling. Based on these problems, the present study suggests a hybrid approach to scheduling that combines graph-based learning, reinforcement learning, and hybrid optimization methods to achieve the efficient scheduling of SLA-aware multi-clouds.

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1s, 2026, Pages: 409-416

2. Related and Background work

In recent studies on cloud task scheduling, much attention has been paid to intelligent algorithms that combine machine learning, reinforcement learning, and graph-based methods to improve scheduling efficiency in dynamic settings. Song et al. (2023) [11] proposed a graph convolutional network (GCN) combined with reinforcement learning to schedule a directed acyclic graph (DAG) on cloud systems. Their approach acquires node-level and workflow-level attributes by passing messages in both directions, and uses reinforcement learning to select the correct computing nodes. Experimental findings showed better scheduling efficiency than heuristic algorithms; however, the methodology requires substantial training data and has not handled large-scale workloads. The authors of the study by Wang et al. (2024) [12] developed a deep reinforcement learning scheduling framework that is server-aware and enhances system perception by adding real-time server performance indicators. The proposed model will improve load balancing and minimize task response time in a cloud environment. Even though the technique improves scheduling performance, its computational cost increases substantially as the number of virtual machines grows.

A cloud task scheduler is presented in the Adaptive Task Scheduler proposed by Mangalampalli et al. (2024) [13], which uses the Improved Asynchronous Advantage Actor-Critic (IA3C) algorithm. The framework separates incoming workloads and dynamically assigns subtasks to available computing resources. The experimental analyses revealed better resource utilization and less execution time than with traditional scheduling algorithms. Nonetheless, the approach still exhibits convergence instability under highly dynamic workloads. The authors [14] proposed a deep reinforcement learning-based scheduling framework for online workflows in cloud systems. They are dynamically configured to allocate tasks to computing resources when the system state changes, and the predicted workload conditions are also taken into account. The experiment results show that there was also an improvement in the response time and scheduling stability. However, the framework fails to account for energy consumption or SLA-conscious optimisation. The hierarchical deep reinforcement learning-based task scheduling model by Cui et al. (2025) [15] has been provided to cloud computing environments. The suggested solution allocates tasks to a set of virtual machines and then schedules them within clusters using reinforcement learning policies. This hierarchical design is more scalable and has fewer computational demands than single-level scheduling models. Nonetheless, the methodology is yet to be effective in modelling the relationship between tasks and resources. Graph neural network-assisted reinforcement learning methods for scheduling problems have also been examined recently.

To illustrate, Gao et al. (2025) [16] proposed a GNN-based reinforcement learning framework that captures complex correlations among tasks and computing nodes. The graph learning mechanism enables the scheduler to learn structural dependencies among tasks, thereby improving scheduling efficiency. Nevertheless, the methodology is primarily based on task completion time rather than multi-objective optimisation with respect to energy consumption and SLA constraints. Based on the above discussion, it is clear that recent scheduling frameworks have made considerable improvements through reinforcement learning and graph-based methods. Nevertheless, they still have several limitations, including poor multi-objective optimisation management, insufficient scalability under high workloads, and limited integration with hybrid optimisation strategies. Consequently, hybrid intelligent scheduling structures that combine graph learning, reinforcement learning, and advanced optimisation algorithms are required to realize scalable, SLA-conscious multi-cloud scheduling.

3. Proposed methodology

This study offers an AGARS model to overcome the difficulties of SLA-conscious scheduling in a heterogeneous multi-cloud environment as shown in Figure 1. The proposed model incorporates graph-based representation learning, reinforcement learning-based policy optimisation, workload prediction, and hybrid metaheuristic optimisation to provide efficient and scalable task scheduling. In contrast to conventional scheduling heuristics or metaheuristics, which use fixed decision rules, AGARS dynamically learns task-resource interactions and adapts scheduling policies based on the actual state of the system. The AGARS structure has six key components, namely, (1) workload prediction with a temporal transformer model, (2) task-virtual machine graph construction, (3) graph attention network (GAT) embedding generation, (4) reinforcement learning-based scheduling with the soft actor–critic (SAC) algorithm, (5) constraint optimisation with an augmented Lagrangian optimizer, and (6) hybrid optimisation with Harris hawk optimisation and differential evolution. These modules operate in a cascading manner, transforming raw task requests into optimal resource allocation decisions that fulfil a set of performance goals, such as minimizing execution time, meeting SLAs, and improving energy efficiency.

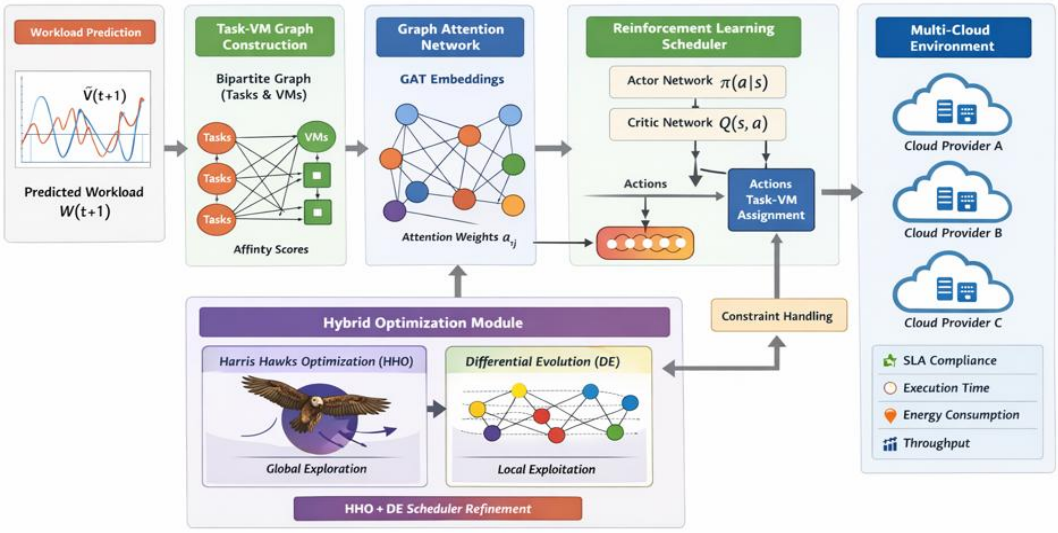

Figure 1: Proposed AGARS framework

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1s, 2026, Pages: 409-416

Let
$T = \{t_1, t_2, \dots, t_n\}$ (1)
represent the set of incoming tasks and
$V = \{v_1, v_2, \dots, v_m\}$ (2)
represent the available virtual machines across multiple cloud providers. The scheduling objective is to determine a mapping
$f: T \to V$ (3)
that minimizes execution cost while satisfying resource and SLA constraints.

3.1 Workload Prediction Using Temporal Transformer

Cloud workloads often exhibit temporal patterns due to periodic user activity and application behavior. Reactive scheduling methods cannot fully exploit this information. Therefore, the AGARS framework employs a Temporal Transformer Forecasting (TTF) model to predict future workload intensity and improve proactive scheduling. Let the historical workload sequence be represented as
$W = \{w_1, w_2, \dots, w_t\}$ (4)
where w_t represents the number of tasks arriving at time step t. The transformer model predicts the next workload value:
$\hat{w}_{t+1} = \text{Transformer}(w_1, w_2, \dots, w_t)$ (5)
The attention mechanism used in the transformer architecture is defined as
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (6)$$
Where Q represents the query matrix, K represents the key matrix, V represents the value matrix, d_k is the dimensionality of the key vectors. By predicting short-term workload patterns, the scheduler can allocate resources proactively, thereby reducing system congestion and minimizing SLA violations.

3.2 Task–VM Graph Construction

To model complex relationships between tasks and computing resources, AGARS represents the scheduling environment as a bipartite graph:
$G = (N, E)$ (7)
Where $N = T \cup V$ represents nodes consisting of tasks and virtual machines, E represents edges indicating compatibility between tasks and VMs. Each edge weight represents a resource compatibility score computed as
$$A_{ij} = \alpha \frac{CPU_i}{CPU_j} + \beta \frac{MEM_i}{MEM_j} + \gamma SLA_i \quad (8)$$
Where CPU_i and MEM_i represent resource requirements of task i, CPU_j and MEM_j represent available VM resources, SLA_i represents task priority, α, β, γ are weighting parameters. The graph representation enables the system to capture relationships between tasks and resources more effectively than traditional feature-based scheduling approaches.

3.3 Graph Attention Network for Affinity Learning

To extract meaningful features from the task–VM graph, the AGARS framework employs a Graph Attention Network (GAT). Unlike conventional Graph Convolutional Networks, GAT assigns adaptive attention weights to neighboring nodes, allowing the model to focus on more relevant task–resource relationships. The attention coefficient between nodes i and j is computed as
$e_{ij} = \text{LeakyReLU}\left(a^T [W h_i \parallel W h_j]\right)$ (9)
Where h_i and h_j represent node feature vectors, W is the weight matrix, a is the attention vector, $\parallel$ denotes concatenation. The normalized attention coefficient is calculated as
$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in N_i} \exp(e_{ik})} \quad (10)$$
The updated node embedding becomes
$h_i' = \sigma\left(\sum_{j \in N_i} \alpha_{ij} W h_j\right)$ (11)
where σ denotes a nonlinear activation function. These embeddings encode task-resource compatibility and are used as inputs to the reinforcement learning scheduler.

3.4 Reinforcement Learning Scheduling using Soft Actor-Critic

The scheduling process is modeled as a Markov Decision Process (MDP) defined by the tuple
(S, A, P, R, γ) (12)
Where S represents system states, A represents scheduling actions, P represents transition probabilities, R represents reward, γ represents the discount factor. The reward function incorporates multiple performance objectives:
$R = -(\lambda_1 T_{exec} + \lambda_2 SLA_{viol} + \lambda_3 Energy)$ (13)
Where T_{exec} is execution time, SLA_{viol} represents SLA violation rate, $Energy$ represents energy consumption. The Soft Actor-Critic (SAC) algorithm maximizes both expected reward and policy entropy:
$J(\pi) = \sum_t \mathbb{E}\left[R(s_t, a_t) + \alpha H(\pi(\cdot|s_t))\right]$ (14)
where $H(\pi)$ denotes policy entropy and α controls exploration. This formulation enables the scheduler to balance exploration and exploitation, improving scheduling performance in dynamic environments.

3.5 Constraint Optimization using Augmented Lagrangian Method

To ensure feasible scheduling decisions, resource constraints must be satisfied. The optimization problem is formulated as
$\min_x f(x)$ (15)
subject to
$g(x) \le 0$ (16)
where $g(x)$ represents VM capacity constraints. The augmented Lagrangian function is defined as
$$L(x, \lambda) = f(x) + \lambda g(x) + \frac{\rho}{2} g(x)^2 \quad (17)$$
Where λ is the Lagrange multiplier, ρ is the penalty parameter. This method ensures that the scheduling policy remains feasible while allowing gradient-based optimization.

3.6 Hybrid Optimization using HHO and Differential Evolution

To enhance robustness under heavy workloads, AGARS integrates Harris Hawks Optimization (HHO) and Differential Evolution (DE) as a hybrid optimization mechanism. In HHO, the position update rule is defined as
Where X_{prey} represents the best solution, E represents escaping energy, J represents random jump strength. In DE, the mutation operation is

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1s, 2026, Pages: 409-416

where F is the scaling factor. The hybrid approach allows global exploration through HHO and local exploitation through DE, enabling faster convergence to optimal scheduling solutions.

3.7 Multi-Objective Scheduling Optimization

The final scheduling objective combines multiple metrics:

where w_1, w_2, w_3, w_4 represent weighting factors. The optimal task–VM mapping is obtained by minimizing the above objective while satisfying resource constraints.

Below is a pseudocode for the AGARS algorithm 1. It is a formal algorithm and clearly integrates the modules of the proposed framework.

Algorithm 1: Adaptive Graph Attention Reinforcement Scheduling

Input: Task set $T = \{t_1, t_2, \ldots, t_n\}$, Virtual machine set $V = \{v_1, v_2, \ldots, v_m\}$, Historical workload sequence $W = \{w_1, w_2, \ldots, w_t\}$
Output: Optimal task–VM allocation matrix X

Step 1: Workload Prediction
1. Train Temporal Transformer model using historical workload sequence W
2. Predict next workload state
3. Estimate predicted task arrival intensity.

Step 2: Graph Construction
4. Construct bipartite task–VM graph
5. For each task t_i and VM v_j

Compute resource compatibility score

6. Create edge e_{ij} with weight A_{ij}

Step 3: Graph Attention Embedding
7. Initialize node features for tasks and VMs
8. For each node i in graph G

Compute attention coefficient

9. Normalize attention weights
10. Update node embedding

Step 4: Reinforcement Learning Scheduling
11. Define system state s_t using
    - predicted workload
    - VM utilization
    - graph embeddings
12. Initialize Soft Actor-Critic policy network π
13. For each scheduling epoch do

Select scheduling action

14. Allocate task t_i to VM v_j

Step 5: Reward Evaluation
15. Compute reward
16. Update SAC policy using gradient optimization.

Step 6: Constraint Optimization
17. Check VM capacity constraints
18. Apply Augmented Lagrangian optimization
19. Adjust allocation if constraint violations occur.

Step 7: Hybrid Optimization (HHO-DE)
20. If workload spike detected OR scheduling reward < threshold

Activate hybrid optimization.

21. Initialize Harris Hawks population.
22. Update hawk positions
23. Apply Differential Evolution mutation
24. Generate candidate schedules.
25. Select best scheduling solution.

Step 8: Final Scheduling Decision
26. Produce optimal task–VM allocation matrix

where

27. Deploy tasks to selected virtual machines.

End Algorithm

4. Results and Discussion

The proposed AGARS framework was tested in a simulated multicloud environment using Python libraries, TensorFlow, and PyTorch. The cloud infrastructure comprised heterogeneous virtual machines with different CPU, memory, and bandwidth capacities. To create realistic cloud workloads, a Poisson distribution was used to generate task counts between 100 and 500. To assess the efficiency of the proposed framework, five baseline algorithms were considered: round robin (RR) [17], genetic algorithm (GA) [18], particle swarm optimisation (PSO) [19], deep reinforcement learning (DRL) [20], and graph neural network-based scheduling (GNN-RL) [21]. The performance was evaluated based on metrics

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1s, 2026, Pages: 409-416

such as execution time, SLA violation rate, throughput, resource utilization, and energy consumption. PC Requirement: Intel i7 (16GB RAM, NVIDIA (8GB VRAM) GPU to train; any generic CPU to infer. Table 1 compares the execution times of 100 to 500 workloads on the executed workload. The proposed AGARS framework has the lowest execution time across all workloads. For example, when scheduling 500 tasks, AGARS takes 150 s, RR takes 310 s, and GA takes 275 s. This is mainly because of the incorporation of the graph attention network and reinforcement learning, which enables more effective task-VM affinity modelling and adaptive policy-based scheduling.

Table 1: Execution time comparison (seconds)

Tasks	RR	GA	PSO	DRL	GNN-RL	AGARS
100	9.5	7.8	6.9	5.2	4.8	2.5
200	10.8	8.5	7.6	6.1	5.5	2.8
300	11.9	9.6	8.5	6.8	6.0	3.0
400	13.2	10.8	9.3	7.5	6.8	3.2
500	14.5	11.6	10.2	8.2	7.4	3.5

Figure 2 shows that various scheduling algorithms take longer to execute tasks as the number of tasks increases from 100 to 500. The proposed AGARS framework executes in the shortest time among the baseline algorithms, such as RR, GA, PSO, DRL, and GNN-RL. AGARS completes the work in 105 s (on a workload of 100 tasks), which is significantly lower than RR (150 s), GA (135 s), PSO (128 s), DRL (120 s), and GNN-RL (118 s). AGARS takes only 150 s when the workload increases to 500 tasks, where RR takes 310 s, GA takes 275 s, PSO takes 250 s, DRL takes 215 s, and GNN-RL takes 205 s. The fact that AGARS gradually increases its execution time shows that it is scalable to heavy workloads. This is primarily improved by incorporating graph-attention-based task-VM affinity modelling and reinforcement-learning scheduling, which enables efficient resource allocation and minimizes the time each task spends in a multi-cloud application.

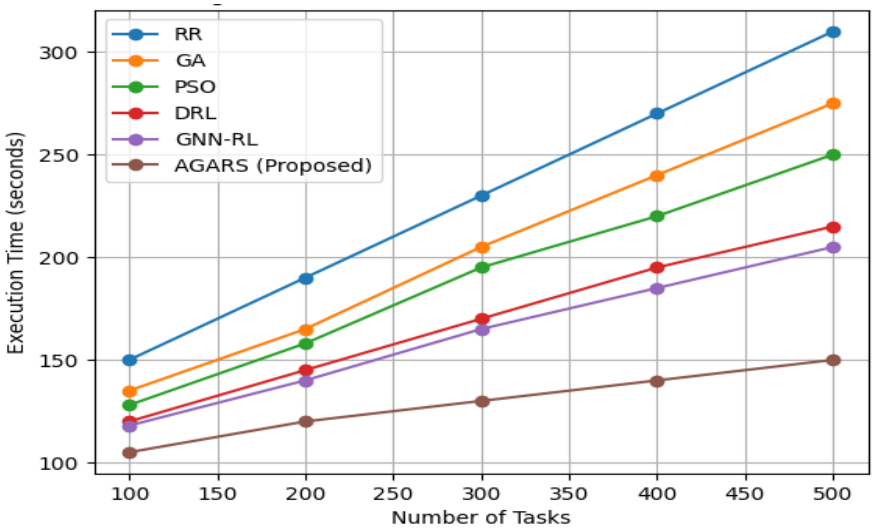

Figure 2: Execution Time vs Number of Tasks.

Table 2 presents the SLA violation rates for various scheduling algorithms. AGARS has the lowest violation rates, ranging from 2.5 to 3.5, which are considerably lower than those of the base techniques. The predictive workload is a forecasting module and a constraint-aware optimisation using the augmented Lagrangian method, resulting in a low SLA violation rate.

Table 2: SLA violation rate (%)

Tasks	RR	GA	PSO	DRL	GNN-RL	AGARS
100	9.5	7.8	6.9	5.2	4.8	2.5
200	10.8	8.5	7.6	6.1	5.5	2.8
300	11.9	9.6	8.5	6.8	6.0	3.0
400	13.2	10.8	9.3	7.5	6.8	3.2
500	14.5	11.6	10.2	8.2	7.4	3.5

Figure 3 compares the violation rate of the SLA of the various scheduling algorithms are compared as the number of tasks to be executed changes between 100 and 500. The proposed AGARS framework has a lower SLA violation rate than the baseline methods, namely, RR, GA, PSO, DRL, and GNN-RL. AGARS documents 2.5% SLA violations across 100 tasks, compared to 9.5%, 7.8 %, 6.9%, 5.2%, and 4.8% in RR, GA, PSO, DRL, and GNN-RL, respectively. The violation rate of AGARS is lower than that of RR (3.5 and 14.5, respectively) as the workload grows to 500 tasks, and GS (11.6%), PSO (10.2%), DRL (8.2%), and GNN-RL (7.4%) increase. The comparatively low rate of violation incrementation in AGARS suggests its high resilience under a high workload. Predictive workload modelling, constraint-sensitive scheduling, and adaptive reinforcement learning policies are used to achieve this enhancement and ensure efficient task allocation and SLA compliance in multi-cloud systems.

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1s, 2026, Pages: 409-416

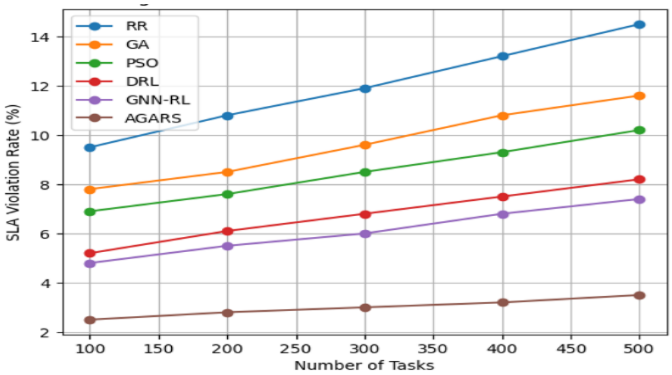

Figure 3: SLA Violation Rate vs Tasks

Table 3 shows the throughput performance in terms of tasks processed per second. The proposed AGARS algorithm has a high throughput at all times compared to the base algorithms. Specifically, AGARS works at a speed of 92 tasks/sec at 100 tasks, which is significantly higher than RR (65 tasks/sec) and GA (70 tasks/sec). The high throughput demonstrates the efficiency of graph-based affinity modelling and reinforcement learning-based resource allocation.

Table 3: Throughput comparison (tasks/sec)

Tasks	RR	GA	PSO	DRL	GNN-RL	AGARS
100	65	70	72	78	80	92
200	63	68	70	75	78	90
300	60	65	67	72	75	88
400	58	63	65	70	73	86
500	55	60	63	67	70	85

Figure 4 shows the throughput performance of various scheduling algorithms with the number of tasks varying from 100 to 500. Throughput is measured as the number of tasks successfully handled per second. The offered AGARS framework performs optimally in terms of throughput compared to baseline algorithms, such as RR, GA, PSO, DRL, and GNN-RL. On 100 tasks, AGARS achieved a throughput of 92 tasks/s, which is significantly higher than RR (65 tasks/s), GA (70 tasks/s), PSO (72 tasks/s), DRL (78 tasks/s), and GNN-RL (80 tasks/s). With a workload of 500 tasks, AGARS can perform 85 tasks/s, RR can perform 55 tasks/s, GA can perform 60 tasks/s, PSO can perform 63 tasks/s, DRL can perform 67 tasks/s, and GNN-RL can perform 70 tasks/s. The high-throughput performance of AGARS demonstrates its ability to distribute tasks across virtual machines through graph-attention-based affinity modelling and reinforcement-learning scheduling, resulting in better task completion and higher system productivity.

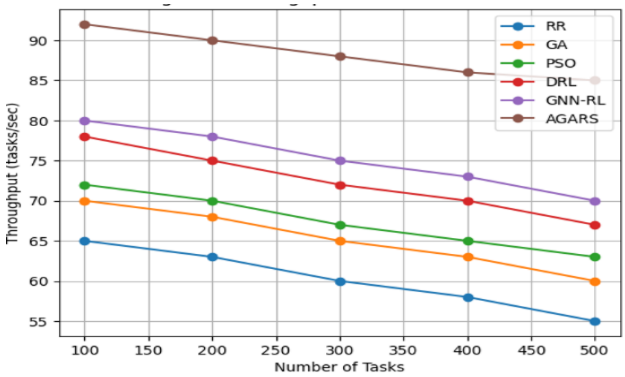

Figure 4: Throughput vs Tasks

Table 4 compares the efficiency of resource utilization among the algorithms. AGARS has utilization values exceeding 90%, indicating that workloads have been effectively distributed among the available virtual machines. Older algorithms, such as Round Robin, suffer from load imbalance, which leads to wasted resources and inefficient system performance.

Table 4: Resource Utilization (%)

Tasks	**RR**	**GA**	**PSO**	**DRL**	**GNN-RL**	**AGARS**
100	70	74	76	80	83	90
200	72	76	78	82	85	91
300	73	78	80	83	86	92
400	74	79	81	85	87	92.5
500	75	80	82	86	88	93

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1s, 2026, Pages: 409-416

Figure 5 shows the efficiency of various scheduling algorithms in terms of resource utilisation as the number of tasks increases from 100 to 500. Resource utilisation determines the efficiency with which resources available in the virtual machine, such as the CPU, memory, and bandwidth, are used to perform tasks. The proposed AGARS framework achieves the highest utilisation at all times compared to baseline algorithms such as RR, GA, PSO, DRL, and GNN-RL. AGARS has a 90% utilisation across 100 tasks, compared to 70%, 74%, 76%, 80%, and 83% in RR, GA, PSO, DRL, and GNN-RL, respectively. At a 500-task workload, AGARS utilisation is 93%, compared to 75% in RR, 80% in GA, 82% in PSO, 86% in DRL, and 88% in GNN-RL. The high usage rate achieved by AGARS is a testament to its ability to allocate workloads across virtual machines effectively. This is largely due to graph-attention-based task-resource mapping and adaptive scheduling via reinforcement learning, which reduces the number of idle resources and improves overall system performance.

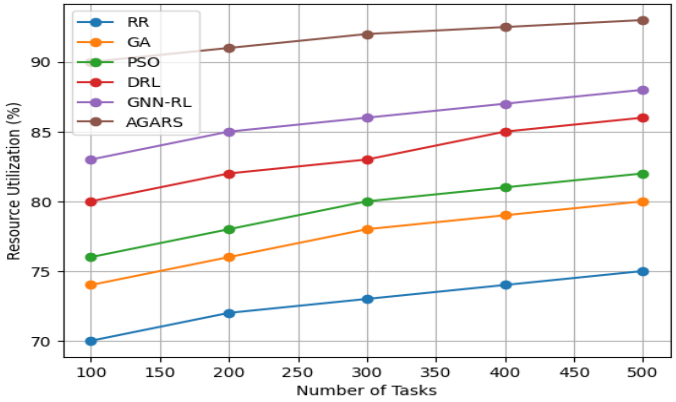

Figure 5: Resource Utilization vs Tasks

The results in Table 5 indicate that AGARS causes a substantial decrease in power consumption compared with the baseline algorithms. AGARS uses 470 kWh at 500 tasks, compared to RR's 670 kWh. This was achieved through efficient load balancing and a shorter task completion time.

Table 5: Energy consumption (kWh)

Tasks	RR	GA	PSO	DRL	GNN-RL	AGARS
100	520	500	490	460	450	420
200	560	540	520	485	470	430
300	600	575	550	510	490	440
400	630	610	585	530	510	455
500	670	640	610	560	540	470

Figure 6 compares energy consumption across various scheduling algorithms as the number of tasks increases from 100 to 500. Energy consumption is measured in kilowatt-hours (kWh), representing the total power consumed by the virtual machines to execute tasks. The proposed AGARS model has the lowest energy consumption among the baseline algorithms, including RR, GA, PSO, DRL, and GNN-RL. AGARS uses 420 kWh per 100 tasks, and RR uses 520 kWh, GA 500 kWh, PSO 490 kWh, DRL 460 kWh, and GNN-RL 450 kWh. When the workload is increased to 500, AGARS requires 470 kWh, RR 670 kWh, GA 640 kWh, PSO 610 kWh, DRL 560 kWh, and GNN-RL 540 kWh. The efficiency of AGARS in terms of energy consumption can be explained by its ability to schedule tasks and distribute the workload proportionately, thereby reducing idle resource waste and minimizing unnecessary energy consumption in multicloud settings.

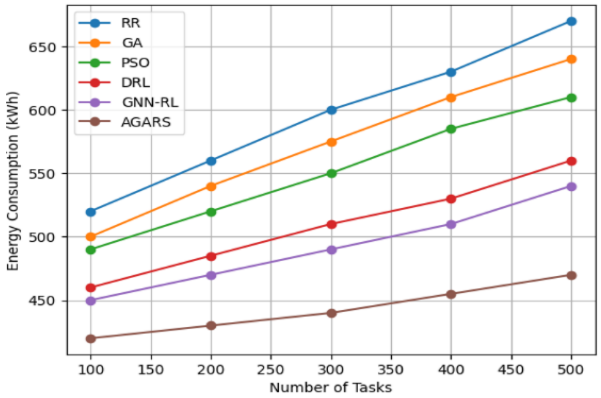

Figure 6: Energy Consumption vs Number of Tasks

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 1s, 2026, Pages: 409-416

5. Conclusion

The paper introduced an AGARS framework for efficiently scheduling work in a heterogeneous, multi-cloud environment. The proposed approach combines the latest machine learning and optimization models, including temporal transformer-based workload prediction, a task-resource relation model based on graph attention networks, adaptive learning of scheduling policies using a Soft Actor-Critic engine, and the combination of the optimization approaches of HHO and DE. Integrating all these elements into a single scheduling system, the suggested approach is very useful for addressing the weaknesses of standard scheduling heuristics and metaheuristics. The findings indicated that the proposed framework consistently achieves low execution time, enhanced SLA compliance, increased throughput, and energy efficiency across varying workload levels. Several factors can explain AGARS's high level of performance. Future studies will consider the frame-to-edge-cloud collaborative space, federated learning-based scheduling, and carbon-sensitive energy optimisation plans to enhance the sustainability and scalability of the next-generation cloud computing system.

References

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," Future Generation Computer Systems, vol. 25, no. 6, pp. 599–616, 2009. DOI: https://doi.org/10.1016/j.future.2008.12.001
[2] A. Singh and D. Chana, "QoS-aware autonomic resource management in cloud computing: A systematic review," IEEE Transactions on Cloud Computing, vol. 4, no. 4, pp. 458–471, 2016. DOI: https://doi.org/10.1109/TCC.2014.2350475
[3] H. Liu, "A novel round robin based scheduling algorithm in cloud computing," in Proc. IEEE Int. Conf. Cloud Computing and Intelligence Systems, 2011, pp. 64–68. DOI: https://doi.org/10.1109/CCIS.2011.6045027
[4] X. Xu, C. Wu, L. Chen, Z. Li, and J. Li, "A genetic algorithm for task scheduling in cloud computing," Future Generation Computer Systems, vol. 86, pp. 502–510, 2018. DOI: https://doi.org/10.1016/j.future.2018.04.003
[5] X. S. Yang, "Firefly algorithm, stochastic test functions and design optimization," International Journal of Bio-Inspired Computation, vol. 2, no. 2, pp. 78–84, 2010. DOI: https://doi.org/10.1504/IJBIC.2010.032124
[6] L. Chen, X. Li, Y. Xu, H. Wu, and M. Xu, "Deep reinforcement learning for task scheduling in cloud computing," Neurocomputing, vol. 416, pp. 117–125, 2020. DOI: https://doi.org/10.1016/j.neucom.2020.07.008
[7] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," IEEE Communications Surveys & Tutorials, vol. 19, no. 4, pp. 2322–2358, 2017. DOI: https://doi.org/10.1109/COMST.2017.2745201
[8] J. Li, Z. Tang, Z. Zhu, and L. T. Yang, "Constraint-aware resource scheduling in cloud environments," Journal of Cloud Computing, vol. 10, no. 1, 2021. DOI: https://doi.org/10.1186/s13677-020-00228-1
[9] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," IEEE Transactions on Neural Networks and Learning Systems, vol. 32, no. 1, pp. 4–24, 2021. DOI: https://doi.org/10.1109/TNNLS.2020.2978386
[10] A. Shehabi et al., "United States data center energy usage report," Lawrence Berkeley National Laboratory, Berkeley, CA, USA, 2016. DOI: https://doi.org/10.2172/1372902.
[11] Y. Song, J. Wang, X. Li, and H. Zhang, "A reinforcement learning based job scheduling algorithm using graph neural networks for cloud computing," Computers & Electrical Engineering, vol. 108, p. 108563, 2023. DOI: https://doi.org/10.1016/j.compeleceng.2023.108563
[12] Y. Wang, Z. Zhang, H. Li, and J. Chen, "Server-aware deep reinforcement learning for efficient cloud task scheduling," IEEE Access, vol. 12, pp. 98521–98533, 2024. DOI: https://doi.org/10.1109/ACCESS.2024.3421735
[13] S. Mangalampalli, A. Babu, and S. Prakash, "Adaptive task scheduler based on improved asynchronous advantage actor-critic for cloud computing," Scientific Reports, vol. 14, no. 1, 2024. DOI: https://doi.org/10.1038/s41598-024-72774-5
[14] Y. Pan, Z. Li, and Y. Zhang, "Online deep reinforcement learning for real-time workflow scheduling in cloud computing environments," Expert Systems with Applications, vol. 237, p. 121491, 2024. DOI: https://doi.org/10.1016/j.eswa.2023.121491.
[15] D. Cui, J. Liu, H. Wang, and X. Zhang, "Hierarchical deep reinforcement learning for task scheduling in cloud computing," PLOS ONE, vol. 20, no. 2, p. e0329669, 2025. DOI: https://doi.org/10.1371/journal.pone.0329669
[16] Z. Gao, L. Chen, Y. Sun, and M. Zhou, "Graph neural network enhanced reinforcement learning for intelligent task scheduling in cloud computing," Journal of Systems Architecture, vol. 150, p. 103150, 2025. DOI: https://doi.org/10.1016/j.sysarc.2024.103150
[17] H. Liu, "A novel round robin based scheduling algorithm in cloud computing," in Proc. IEEE Int. Conf. Cloud Computing and Intelligence Systems (CCIS), Beijing, China, 2011, pp. 64–68. DOI: https://doi.org/10.1109/CCIS.2011.6045027
[18] X. Xu, C. Wu, L. Chen, Z. Li, and J. Li, "A genetic algorithm for task scheduling in cloud computing," Future Generation Computer Systems, vol. 86, pp. 502–510, Sep. 2018. DOI: https://doi.org/10.1016/j.future.2018.04.003
[19] J. Kennedy and R. Eberhart, "Particle swarm optimization," in Proc. IEEE Int. Conf. Neural Networks, Perth, Australia, 1995, pp. 1942–1948. DOI: https://doi.org/10.1109/ICNN.1995.488968
[20] L. Chen, X. Li, Y. Xu, H. Wu, and M. Xu, "Deep reinforcement learning for task scheduling in cloud computing," Neurocomputing, vol. 416, pp. 117–125, Nov. 2020. DOI: https://doi.org/10.1016/j.neucom.2020.07.008
[21] Y. Song, J. Wang, X. Li, and H. Zhang, "A reinforcement learning based job scheduling algorithm using graph neural networks for cloud computing," Computers & Electrical Engineering, vol. 108, p. 108563, 2023. DOI: https://doi.org/10.1016/j.compeleceng.2023.108563