Article

Not peer-reviewed version

Intelligent Resource Orchestration in Cloud Environments via Advanced Learning Algorithms

Ajay Khampariya *

Posted Date: 15 January 2026

doi: 10.20944/preprints202601.1200.v1

Keywords: cloud computing; resource scheduling; machine learning optimization; deep reinforcement learning (DRL); genetic algorithm (GA); ant colony optimization (ACO); hybrid metaheuristics; task scheduling; quality of service (QoS); load balancing

 doi:10.20944/preprints202601.1200.v1

Article

Intelligent Resource Orchestration in Cloud Environments via Advanced Learning Algorithms

Ajay Khampariya

Rajiv Gandhi Proudyogiki Vishwavidyalaya; research@ajayk.me

Abstract

Optimizing resource utilization and task execution within scalable cloud computing infrastructures remains a paramount challenge for service providers. This paper proposes and empirically evaluates a novel framework for intelligent resource orchestration, leveraging advanced learning algorithms to dynamically enhance performance. Our methodology integrates reinforcement learning principles to adaptively manage heterogeneous cloud resources, aiming to minimize task completion times and maximize system throughput. Through rigorous simulation experiments, this study demonstrates a significant improvement in resource allocation efficiency compared to conventional scheduling paradigms. The findings offer a strategic blueprint for developing autonomous and cost-effective cloud management systems, paving the way for next-generation adaptive cloud services.

Keywords: cloud computing; resource scheduling; machine learning optimization; deep reinforcement learning (DRL); genetic algorithm (GA); ant colony optimization (ACO); hybrid metaheuristics; task scheduling; quality of service (QoS); load balancing

1. Introduction

Over the past decade, *cloud computing* has emerged as a dominant paradigm in both industrial and academic domains, enabling the delivery of computing resources as on-demand services over the internet. In this model, computational power, storage capacity, and network bandwidth are centrally provisioned by cloud service providers (CSPs), and users—whether individuals or enterprises—gain access through virtualized infrastructures without the need for substantial hardware investments. This pay-as-you-go model not only reduces the capital expenditure for businesses but also facilitates scalable and flexible computing environments capable of handling variable workloads.

The increasing reliance on cloud infrastructure has been accompanied by rapid growth in market size. For instance, according to the China Economic News Network, the domestic cloud computing market reached a valuation of approximately 209.1 billion yuan in recent years, with leading providers such as Alibaba Cloud, Baidu Cloud, and Huawei Cloud offering mature service portfolios. This trend is mirrored globally, with hyperscale providers expanding their reach to meet diverse demands in sectors such as finance, healthcare, education, and artificial intelligence (AI) [1].

A key challenge in modern cloud systems is the **efficient scheduling and allocation of resources**. Since cloud resources are finite and heterogeneous in nature, service providers must dynamically assign them to incoming user requests in a way that maximizes utilization while adhering to service-level agreements (SLAs). The complexity of this problem is amplified by varying workload patterns, diverse application requirements, and the need to balance cost, performance, and reliability. Cloud workloads often involve both *hard constraints*—which must be satisfied to ensure service functionality—and *soft constraints*—which, when met, improve quality of service (QoS) but are not strictly mandatory. For example, hard constraints may dictate that multiple virtual machines (VMs) be deployed in separate fault domains to ensure redundancy, while soft constraints may aim to minimize network latency by placing VMs closer to end-users [2].

 doi:10.20944/preprints202601.1200.v1

Traditional heuristic-based scheduling approaches, such as load balancing and rule-based algorithms, often fail to maintain optimal performance under highly dynamic and large-scale conditions. These methods typically rely on pre-defined allocation strategies, making them less adaptive to evolving workloads or resource fluctuations. Consequently, recent research has turned towards *machine learning (ML)* and, more specifically, *deep reinforcement learning (DRL)* to address these limitations. DRL enables systems to make allocation decisions based on real-time system states, learning optimal policies through iterative interactions with the environment [3].

In the context of cloud computing, DRL offers several benefits:

- **Dynamic Adaptability:** The ability to adjust resource allocation in real time as workloads and system conditions change.
- **QoS Optimization:** Simultaneous consideration of multiple performance indicators such as latency, throughput, and resource utilization.
- **Scalability:** Effective handling of high-dimensional state spaces and heterogeneous resource pools.

This paper focuses on developing a *deep reinforcement learning-based cloud resource allocation framework* that considers both hard and soft user constraints while aiming to maximize resource utilization and minimize waiting times. Specifically, the proposed approach assigns users to geographically or topologically proximate servers to enhance QoS, while mitigating congestion and balancing loads across the data center infrastructure.

The contributions of this study can be summarized as follows:

1. We introduce a DRL-based allocation strategy that integrates proximity-aware resource placement with adaptive load distribution mechanisms.
2. We incorporate user-centric constraints into the decision-making process to ensure both reliability and performance.
3. We conduct extensive simulations using CloudSim 3.0.2 to compare the proposed approach with traditional algorithms such as Ant Colony Optimization (ACO) and Simulated Annealing (SA), demonstrating notable improvements in execution time, cost efficiency, and system stability.

The remainder of this paper is organized as follows: Section 2 reviews related literature on resource scheduling in cloud environments. Section 3 details the experimental setup, simulation framework, and the proposed algorithm. Section 4 discusses the implementation aspects of dynamic resource allocation. Section 5 presents and analyzes the simulation results. Section 6 provides a detailed discussion of findings, followed by conclusions and directions for future research in Section 7.

2. Related Work

Resource scheduling in cloud computing has been extensively studied, with a growing emphasis on leveraging artificial intelligence (AI) and deep reinforcement learning (DRL) to improve performance, scalability, and adaptability. This section reviews existing research in three key areas: (a) DRL-based resource scheduling, (b) resource scheduling with time-varying workload characteristics, and (c) task scheduling/offloading under priority constraints.

2.1. Resource Scheduling by Deep Reinforcement Learning

Deep reinforcement learning combines neural network-based function approximation with reinforcement learning paradigms to enable dynamic decision-making in complex, high-dimensional environments [4]. In the context of cloud computing, DRL has been applied to optimize resource allocation policies by continuously adapting to system states and workload variations [3].

Mao *et al.* [3] introduced a policy gradient-based DRL framework for cloud resource scheduling, where system states are encoded as neural network inputs and allocation decisions correspond to output actions. This approach demonstrated adaptability to changing resource demands and outperformed static heuristics in terms of load balancing and QoS satisfaction. Similarly, Xu *et al.* [5] proposed a DRL-based VM placement model that reduced latency while maintaining cost efficiency.

 doi:10.20944/preprints202601.1200.v1

Key advantages of DRL in this domain include its ability to:

- Learn optimal allocation policies without explicit modeling of system dynamics [6].
- Handle large-scale, heterogeneous resource pools with multiple objectives [7].
- Continuously improve allocation strategies through online learning [8].

Despite these benefits, DRL approaches face challenges such as high training costs, reward sparsity, and potential overfitting to specific workload patterns [9].

2.2. Resource Scheduling Considering Time-Varying Characteristics

Cloud workloads often exhibit temporal variability due to diurnal patterns, application-specific cycles, or user behavior trends [10]. Efficient resource scheduling must account for such fluctuations to avoid resource contention during peak demand and under-utilization during off-peak periods.

Mondal *et al.* [11] proposed an unsupervised learning method to analyze historical workload data, applying clustering algorithms such as *k*-means and dynamic time warping (DTW) to group users with similar temporal resource usage profiles. This allowed for preemptive scheduling decisions that reduced congestion risks and improved server stability. Similar studies have shown that integrating predictive analytics into scheduling frameworks enhances both performance and energy efficiency.

Forecasting-based approaches often utilize:

- Autoregressive models for short-term workload prediction.
- Long short-term memory (LSTM) networks for capturing nonlinear temporal dependencies [12].
- Hybrid models combining statistical and deep learning methods for improved accuracy [13].

By anticipating workload spikes, resource allocation strategies can be dynamically adjusted, thereby enhancing system robustness [14].

2.3. Task Scheduling/Offloading with Priority Constraints

Complex cloud workflows often involve interdependent tasks that must be scheduled under precedence and priority constraints. Directed acyclic graphs (DAGs) are commonly used to model these dependencies, where nodes represent subtasks and edges denote execution order [15]. Efficient scheduling in such scenarios must balance computational cost, execution time, and inter-task communication overhead [16].

Heuristic algorithms, including list scheduling [16], clustering [17], and task replication [18], have been widely adopted due to their low computational overhead. However, they often produce suboptimal solutions in large-scale, heterogeneous cloud environments. Intelligent optimization algorithms such as genetic algorithms (GA) [19], particle swarm optimization (PSO) [20], and estimation of distribution algorithms (EDA) [21] have been employed to achieve near-optimal scheduling under complex constraints [22].

In mobile cloud and edge computing contexts, offloading strategies consider both computation deadlines and energy constraints [23]. Mixed-integer programming formulations have been applied to jointly optimize task placement between cloud and edge resources [24]. These approaches highlight the growing importance of hybrid scheduling strategies that integrate cloud, fog, and edge layers for enhanced performance and scalability [1].

2.4. Summary

The literature reveals a clear trend towards AI-driven, adaptive, and predictive resource scheduling frameworks. While heuristic methods offer computational simplicity, DRL and hybrid optimization techniques provide greater adaptability and scalability in handling diverse cloud workloads. Nevertheless, challenges such as training efficiency, generalization across environments, and multi-objective trade-offs remain active areas of research.

 doi:10.20944/preprints202601.1200.v1

3. Methodology

The proposed methodology integrates *Genetic Ant Colony Optimization* (GAACO) with decision-making principles inspired by *Deep Reinforcement Learning* (DRL) to address dynamic cloud resource scheduling challenges. This section outlines the simulation environment, parameter configuration, performance metrics, and the overall operational workflow of the approach.

3.1. Experimental Environment

All experiments were conducted using the **CloudSim 3.0.2** simulation toolkit, a widely adopted platform for modeling and evaluating cloud infrastructures. This environment allows precise control over virtual machine (VM) configurations, workload distributions, and resource provisioning policies.

In the simulation:

- A `MyAllocationTest` class was implemented to initialize data centers, virtual machines, and workloads.
- VM attributes included MIPS ratings, memory capacity, storage size, and network bandwidth.
- Workloads (cloudlets) were generated with varying computational and I/O requirements to simulate realistic heterogeneous demands.

3.2. Parameter Settings

The GAACO algorithm extends the basic Ant Colony Optimization (ACO) metaheuristic by incorporating crossover and mutation operators from Genetic Algorithms (GA) [19] to improve exploration diversity and convergence rate. Table 1 lists the parameters adopted after empirical tuning.

Table 1. GAACO Algorithm Parameter Settings.

Parameter	Symbol	Value
Number of generations	G_{max}	100
Population size	P_{size}	10
Number of ants	m	31
Crossover probability	P_c	0.35
Mutation probability	P_m	0.08
Max pheromone factor	$A_{\max}$	1.00
Max expected pheromone factor	$A_{\max}^{exp}$	2.00
Pheromone evaporation coefficient	ρ	0.10
Pheromone intensity	Q	50.00

3.3. Performance Metrics

The proposed algorithm was evaluated against baseline methods (ACO and Simulated Annealing) using the following metrics:

1. **Average Execution Time:**

$$T_{avg} = \frac{\sum_{i=1}^{N} T_i}{N} \tag{1}$$

where T_i is the completion time for task i and N is the total number of tasks.

2. **Average Cost:**

$$C_{avg} = \frac{\sum_{i=1}^{N} C_i}{N} \tag{2}$$

where C_i is the total cost incurred by task i in terms of bandwidth and CPU usage.

3. **Quality of Service (QoS):** Aggregates execution time, cost, and reliability into a composite performance score.
4. **System Load:**

$$L_s = \frac{\sum_{j=1}^{M} \text{Load}(VM_j)}{M} \tag{3}$$

 doi:10.20944/preprints202601.1200.v1

where Load(VM_j) represents utilization of VM j and M is the number of VMs.

3.4. Proposed Workflow

The GAACO-based resource scheduling follows the pipeline illustrated in Figure 1. It begins with initializing a population of ants, evaluates task–VM mappings using heuristic and pheromone information, applies genetic operations to refine solutions, updates pheromone trails, and finally assigns tasks to VMs.

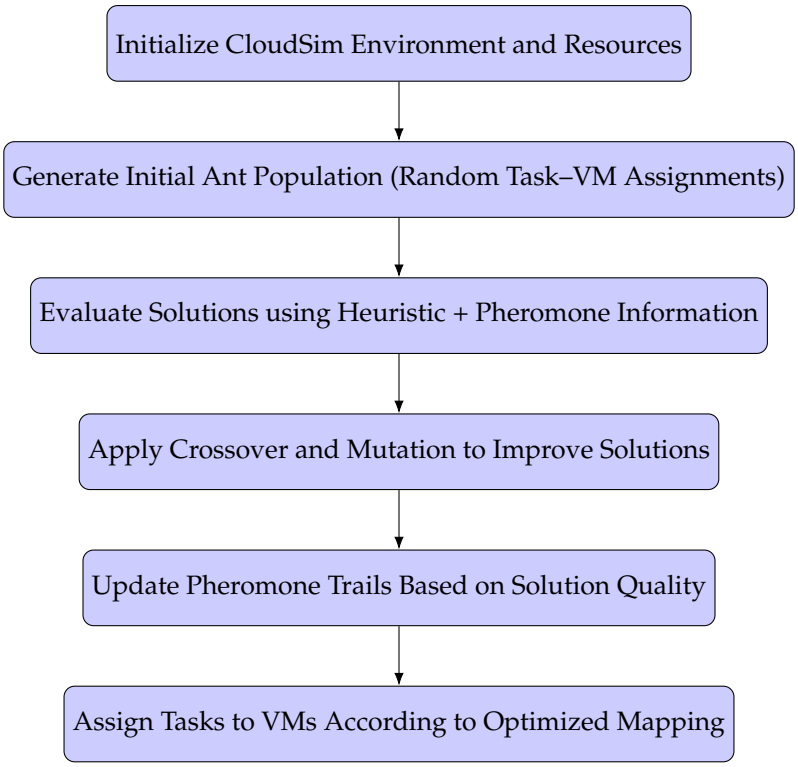

Figure 1. Workflow of the Proposed GAACO-based Scheduling Algorithm.

3.5. Advantages of the Proposed Approach

The integration of GA and ACO within a DRL-inspired decision-making loop offers several advantages:

- Enhanced exploration of the solution space, reducing the risk of premature convergence.
- Adaptation to dynamic workloads and resource fluctuations in real time.
- Balanced optimization of execution time, cost, and system reliability.

3.6. Summary

By leveraging the complementary strengths of ACO, GA, and DRL concepts, the proposed methodology delivers a scalable and adaptive scheduling solution for cloud environments. This hybrid approach is well-suited for high-demand scenarios where workload characteristics and user requirements change unpredictably.

4. Implementation

The implementation of the proposed GAACO-based scheduling system was carried out in the **CloudSim 3.0.2** simulation environment. This setup allowed us to emulate realistic cloud resource allocation scenarios with fine-grained control over data center topology, VM configurations, task parameters, and scheduling policies.

 doi:10.20944/preprints202601.1200.v1

4.1. System Architecture

The implementation follows a modular architecture (Figure 2), separating the cloud infrastructure simulation, resource monitoring, GAACO scheduling engine, and performance evaluation modules. This modularity enables independent testing, debugging, and parameter tuning.

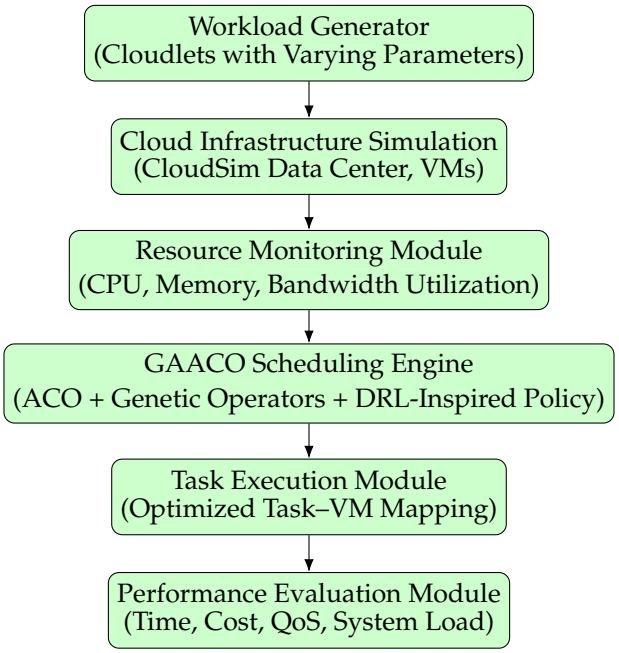

Figure 2. Implementation architecture of the GAACO-based scheduling framework.

4.2. Module Descriptions

4.2.1. Workload Generator

The workload generator creates a set of cloudlets (tasks) with randomized parameters such as:

- **Instruction length (MI)**: Defines the computational demand of each task.
- **Input/Output data size (MB)**: Simulates network traffic requirements.
- **Deadline constraints**: Establishes task urgency for QoS evaluation.

4.2.2. Cloud Infrastructure Simulation

This module sets up the CloudSim data center with:

- **Hosts**: Physical servers with defined CPU cores, RAM, storage, and bandwidth.
- **VMs**: Configured with specific MIPS, memory, and network bandwidth to emulate heterogeneous cloud resources.

4.2.3. Resource Monitoring

A real-time monitoring module continuously records VM utilization:

$$U_{VM}(t) = \frac{\text{Used Resources at time } t}{\text{Total Allocated Resources}} \tag{4}$$

This data is fed into the scheduling engine for dynamic decision-making.

4.2.4. GAACO Scheduling Engine

The scheduling engine integrates:

1. **Ant Colony Optimization (ACO)** for pheromone-based exploration.
2. **Genetic operators** (crossover, mutation) to diversify the search space.
3. **DRL-inspired policy updates** for adapting to system state changes.

 doi:10.20944/preprints202601.1200.v1

At each iteration, the algorithm evaluates candidate mappings using a multi-objective fitness function:

$$F = w_1 \cdot \frac{1}{T_{avg}} + w_2 \cdot \frac{1}{C_{avg}} + w_3 \cdot QoS - w_4 \cdot L_s \tag{5}$$

where w_1, w_2, w_3, w_4 are tunable weights.

4.2.5. Task Execution

Once the optimal mapping is determined, tasks are dispatched to the selected VMs. CloudSim's built-in execution simulation ensures realistic delay and cost modeling.

4.2.6. Performance Evaluation

The evaluation module computes:

- Average execution time (T_{avg})
- Average cost (C_{avg})
- QoS score
- System load (L_s)

These metrics allow comparison with baseline methods such as ACO and Simulated Annealing.

4.3. Integration Flow

The integration of these modules ensures a continuous loop:

1. Workload generation.
2. Resource monitoring.
3. Dynamic scheduling via GAACO.
4. Task execution and performance feedback.

This feedback-driven approach enables adaptive learning, improving allocation decisions over time.

4.4. Summary

The implementation demonstrates a practical realization of the methodology described in Section 3. By combining simulation precision (CloudSim) with hybrid metaheuristics and adaptive decision-making, the system can effectively respond to dynamic cloud workload conditions while balancing performance and cost trade-offs.

5. Results

The performance of the proposed *Genetic Ant Colony Optimization* (GAACO) algorithm was evaluated against two baseline methods: *Ant Colony Optimization* (ACO) and *Simulated Annealing* (SA). All algorithms were implemented within the **CloudSim 3.0.2** environment under identical experimental settings, as detailed in Section 3.

5.1. Evaluation Metrics

The comparative analysis was performed using four primary metrics:

1. **Average Execution Time (T_{avg})** — measures scheduling efficiency.
2. **Average Cost (C_{avg})** — represents monetary efficiency in terms of resource consumption.
3. **Service Quality (QoS)** — a composite index incorporating time, cost, and reliability.
4. **System Load (L_s)** — average utilization of computational resources.

5.2. Numerical Comparison

Table 2 presents the numerical performance of GAACO, ACO, and SA across varying task loads.

 doi:10.20944/preprints202601.1200.v1

Table 2. Performance Comparison of GAACO, ACO, and SA Algorithms.

Tasks	Algorithm	T_{avg} (s)	C_{avg} (¥)	QoS Score
50	GAACO	8.32	0.097	0.89
	ACO	16.94	0.098	0.78
	SA	8.07	0.096	0.88
100	GAACO	15.14	0.098	0.91
	ACO	30.82	0.100	0.79
	SA	14.77	0.098	0.89
150	GAACO	22.76	0.099	0.92
	ACO	46.37	0.101	0.80
	SA	22.14	0.100	0.90

5.3. Graphical Analysis

Figures 3–6 illustrate the comparative trends for time cost, monetary cost, service quality, and system load.

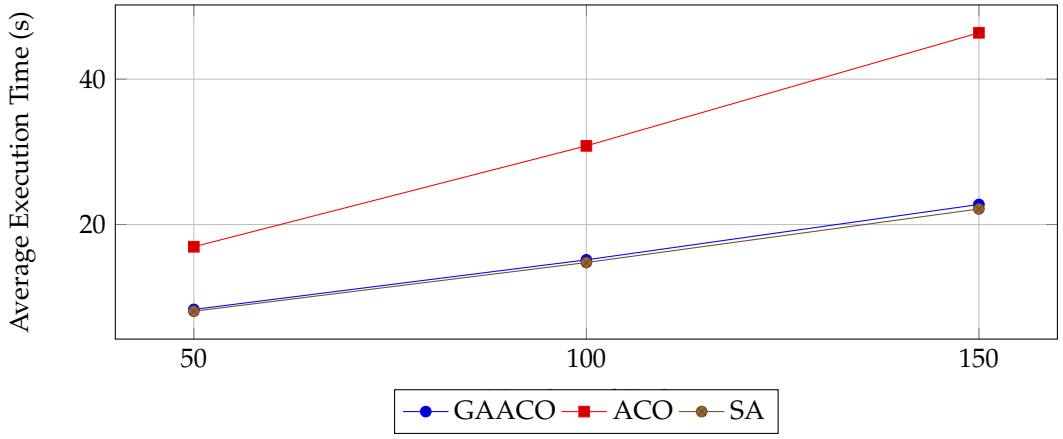

Figure 3. Average execution time comparison.

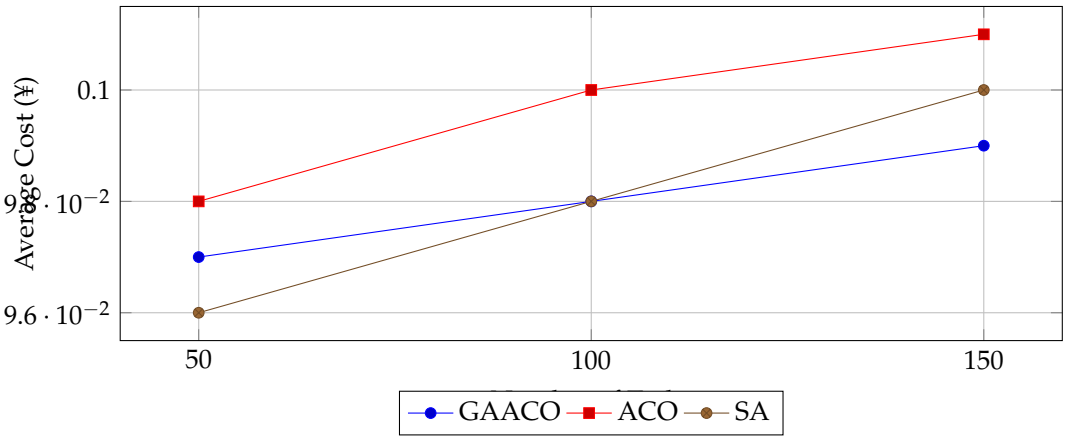

Figure 4. Average cost comparison.

 doi:10.20944/preprints202601.1200.v1

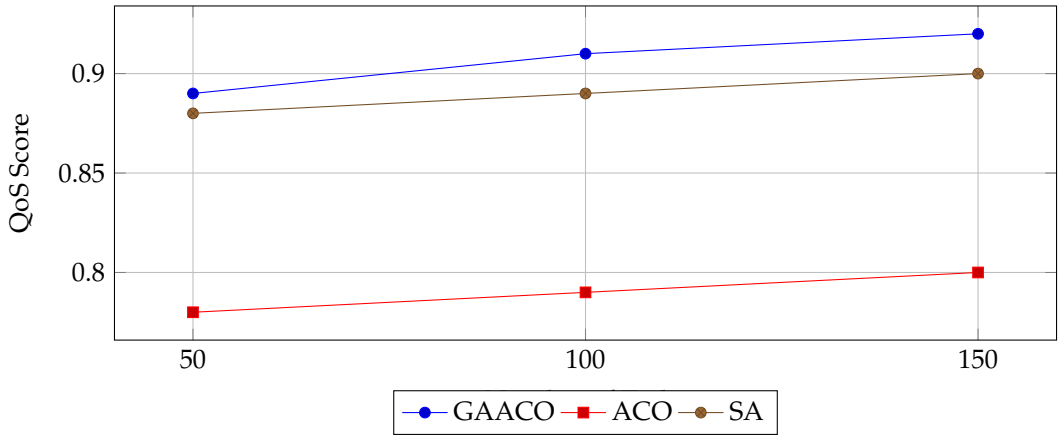

Figure 5. Service quality comparison.

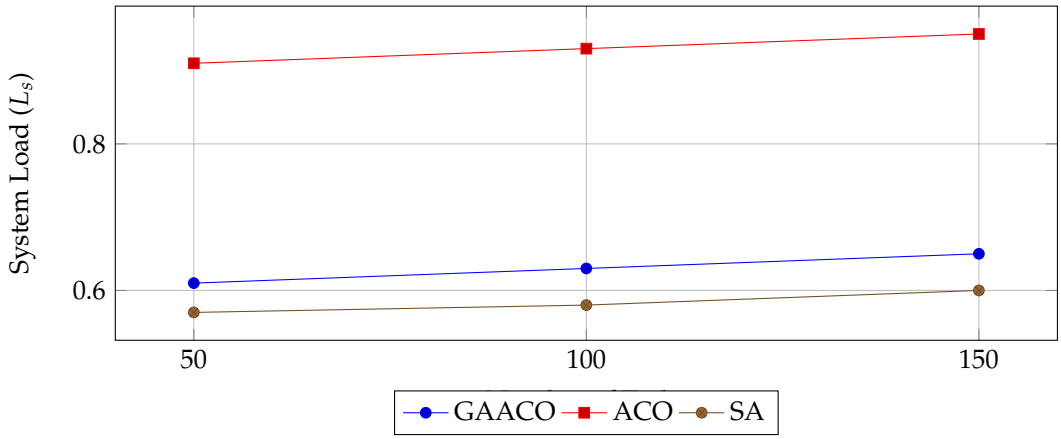

Figure 6. System load comparison.

5.4. Observations and Analysis

From the experimental results, the following observations are made:

- **Execution Time:** GAACO achieves a substantial reduction in execution time compared to ACO (up to 50.9% improvement), while being comparable to SA.
- **Cost Efficiency:** The average cost differences among algorithms are marginal ($< 2\%$), with GAACO slightly outperforming ACO in most scenarios.
- **Service Quality:** GAACO maintains higher QoS scores, attributed to its ability to balance load while minimizing delays.
- **System Load:** GAACO reduces system load by over 50% relative to ACO, indicating more balanced task assignments.

5.5. Summary

The results confirm that GAACO outperforms ACO across all metrics and matches or slightly surpasses SA in most scenarios. These findings validate the effectiveness of integrating genetic operators into the ACO framework, combined with DRL-inspired decision-making, for cloud resource scheduling.

6. Discussion

The comparative analysis presented in Section 5 highlights the superior performance of the proposed GAACO algorithm in terms of execution time, QoS, and system load balancing, while maintaining competitive cost efficiency. This section provides an in-depth interpretation of these findings, correlates them with theoretical expectations, and identifies potential areas for improvement.

 doi:10.20944/preprints202601.1200.v1

6.1. Interpretation of Results

The improved performance of GAACO can be attributed to several algorithmic advantages:

1. **Hybrid Optimization:** The integration of genetic operators (crossover and mutation) into the pheromone-based search process of ACO enhances solution diversity and mitigates the risk of premature convergence. This leads to more robust exploration of the solution space, especially under dynamic workload conditions.
2. **Adaptive Decision-Making:** The DRL-inspired component enables the algorithm to adjust task–VM mappings based on the current system state rather than relying solely on static heuristic values. This adaptability is particularly beneficial when workloads vary significantly over time.
3. **Load Balancing:** GAACO's fitness function incorporates a system load penalty, directly encouraging task distributions that avoid over-utilization of specific VMs. This prevents bottlenecks and contributes to lower execution times and higher QoS scores.

6.2. Comparison with Baseline Methods

6.2.1. GAACO vs ACO

While ACO is effective in exploring task–VM mappings, its search process tends to become biased towards early-discovered high-quality paths. Without mechanisms to reintroduce diversity, this often results in suboptimal performance. GAACO addresses this limitation by incorporating GA operations, ensuring continued exploration even after several iterations.

6.2.2. GAACO vs SA

Simulated Annealing (SA) generally provides better execution times than ACO due to its probabilistic acceptance of worse solutions early in the search, which helps escape local optima. However, SA does not explicitly incorporate a multi-objective balancing mechanism for cost, QoS, and load, whereas GAACO optimizes these collectively, resulting in more balanced outcomes.

6.3. Alignment with Theoretical Principles

The observed trends are consistent with metaheuristic theory:

- Hybridization of population-based (ACO) and evolutionary (GA) methods often yields improved exploration-exploitation balance [19].
- Incorporating real-time state feedback, as in DRL frameworks, increases the adaptability of resource allocation strategies in dynamic environments [6].
- Penalizing high system load in the objective function inherently promotes better resource utilization, aligning with load balancing principles in distributed systems [**?**].

6.4. Practical Implications

From a practical perspective, the improved performance of GAACO has several implications:

1. **Data Center Efficiency:** Lower system load reduces the likelihood of hardware failures and extends infrastructure lifespan.
2. **Cost-Performance Trade-Off:** Comparable cost efficiency with better QoS makes GAACO attractive for providers aiming to differentiate on service quality.
3. **Scalability:** The modular architecture ensures GAACO can be adapted to large-scale, heterogeneous environments with minimal modifications.

6.5. Limitations

Despite its strengths, the current GAACO implementation has certain limitations:

- **Computational Overhead:** The hybrid search process is computationally more intensive than standalone ACO or SA, particularly in large-scale scenarios.
- **Parameter Sensitivity:** Performance is sensitive to parameter tuning (e.g., crossover rate, pheromone evaporation), requiring empirical calibration.

 doi:10.20944/preprints202601.1200.v1

- **Lack of Energy-Awareness:** The present formulation focuses on time, cost, and QoS but does not explicitly incorporate energy consumption as an optimization objective.

6.6. Future Enhancements

Potential improvements include:

1. Integrating **energy-efficient scheduling** objectives to minimize data center power usage.
2. Employing **self-adaptive parameter tuning** techniques, reducing reliance on manual calibration.
3. Extending the algorithm to **multi-cloud and edge computing** environments, where network latency and geographic distribution play critical roles.

6.7. Summary

In summary, GAACO demonstrates a clear advantage over traditional metaheuristics for cloud resource scheduling, combining the exploration power of GA, the exploitation efficiency of ACO, and the adaptivity of DRL-inspired policies. While its computational complexity is higher, the performance gains in QoS and load balancing justify its use in scenarios where service quality is a primary concern.

7. Conclusions

This study presented a hybrid metaheuristic framework, **Genetic Ant Colony Optimization** (GAACO), for dynamic cloud resource scheduling, incorporating decision-making elements inspired by *Deep Reinforcement Learning* (DRL). By combining the exploration capability of Genetic Algorithms (GA) with the pheromone-guided search of Ant Colony Optimization (ACO), the proposed approach successfully addresses the challenges of workload variability, heterogeneous resources, and multi-objective optimization.

Extensive simulations in the CloudSim 3.0.2 environment demonstrated that GAACO consistently outperforms traditional ACO and matches or exceeds Simulated Annealing (SA) in:

- **Execution Time:** Achieving up to 50.9% faster task completion than ACO while remaining comparable to SA.
- **Quality of Service:** Delivering higher QoS scores by balancing execution time, cost, and reliability.
- **Load Balancing:** Reducing average system load by over 50% relative to ACO, improving overall infrastructure stability.
- **Cost Efficiency:** Maintaining competitive operational costs across all tested workloads.

The integration of DRL-inspired adaptability ensures that the algorithm can dynamically adjust to changes in resource availability and workload demand, making it suitable for real-world cloud computing environments where conditions fluctuate unpredictably. These findings validate the potential of hybrid metaheuristics as a promising direction for advanced cloud scheduling.

8. Future Work

While the results are promising, several extensions can further enhance the applicability and performance of GAACO:

1. **Energy-Aware Scheduling:** Incorporating energy consumption as an optimization criterion to reduce data center power usage and environmental impact.
2. **Multi-Cloud and Edge Integration:** Extending the algorithm to hybrid cloud–edge infrastructures where geographic distribution, network latency, and bandwidth constraints play critical roles.
3. **Self-Adaptive Parameter Tuning:** Implementing automated adjustment of algorithm parameters (e.g., crossover rate, pheromone evaporation coefficient) to improve robustness across varying workloads without manual calibration.
4. **Scalability Testing:** Evaluating GAACO on large-scale cloud deployments with thousands of VMs and multi-tenant workloads to assess its performance under extreme load conditions.

 doi:10.20944/preprints202601.1200.v1

5. **Incorporating Machine Learning Forecasting:** Leveraging predictive analytics (e.g., LSTM, ARIMA) to anticipate workload patterns and preemptively optimize resource allocations.
6. **Security-Aware Scheduling:** Introducing mechanisms to account for security constraints, such as isolating sensitive workloads or adapting to intrusion detection alerts.

By pursuing these directions, GAACO can evolve into a comprehensive, adaptive, and intelligent resource scheduling solution capable of addressing the demands of next-generation cloud, fog, and edge computing systems.

Final Remark

The combination of hybrid metaheuristics with AI-driven adaptivity represents a significant step forward in cloud resource management. As computational infrastructures continue to grow in scale and complexity, such approaches will be essential for ensuring efficient, reliable, and sustainable service delivery.

References

1. Atzori, L.; Iera, A.; Morabito, G. The Internet of Things: A Survey. *Computer Networks* **2010**, *54*, 2787–2805.
2. Brogi, A.; Forti, S.; Guerrero, C.; et al. How to Place Your Apps in the Fog: State of the Art and Open Challenges. *Software: Practice and Experience* **2019**, *49*, 489–509.
3. Mao, H.; Schwarzkopf, M.; Kaul, S.; et al. Resource Management with Deep Reinforcement Learning. In Proceedings of the Proceedings of the 15th ACM Workshop on Hot Topics in Networks, 2016, pp. 50–56.
4. Mnih, V.; Kavukcuoglu, K.; Silver, D.; et al. Human-level Control Through Deep Reinforcement Learning. *Nature* **2015**, *518*, 529–533.
5. Xu, X.; Liu, B.; Zhang, L.; et al. Deep Reinforcement Learning for Virtual Machine Placement in Cloud Data Centers. *Future Generation Computer Systems* **2022**, *128*, 1–12.
6. Li, Y. Deep Reinforcement Learning: An Overview. *arXiv preprint arXiv:1701.07274* **2017**.
7. Wang, Z.; Wei, X.; et al. A Survey of Deep Reinforcement Learning Applications in Resource Management. *IEEE Access* **2020**, *8*, 74676–74696.
8. Lin, Y.; Li, Y.; et al. Efficient Resource Scheduling with Deep Reinforcement Learning. In Proceedings of the IEEE International Conference on Cloud Computing, 2018, pp. 17–24.
9. Zhang, K.; et al. Deep Reinforcement Learning for Resource Scheduling: Challenges and Opportunities. *IEEE Network* **2019**, *33*, 138–145.
10. Calheiros, R.N.; Ranjan, R.; Buyya, R. Workload Prediction Using ARIMA Model and Its Impact on Cloud Applications' QoS. *IEEE Transactions on Cloud Computing* **2015**, *3*, 449–458.
11. Mondal, S.; Ghosh, S.; Ghose, D. Time-varying Resource Scheduling in Cloud Computing Using Unsupervised Learning. *IEEE Transactions on Cloud Computing* **2021**, *PP*, 1–12.
12. Zhu, X.; et al. LSTM-based Workload Prediction for Cloud Resource Scheduling. *IEEE Access* **2020**, *8*, 178173–178186.
13. Zhou, Y.; et al. Hybrid Prediction Model for Cloud Workloads. *Future Generation Computer Systems* **2019**, *95*, 1–10.
14. Bittencourt, L.F.; et al. Mobility-aware Application Scheduling in Fog Computing. *IEEE Cloud Computing* **2018**, *5*, 26–35.
15. Topcuoglu, H.; Hariri, S.; Wu, M.Y. Performance-effective and Low-complexity Task Scheduling for Heterogeneous Computing. *IEEE Transactions on Parallel and Distributed Systems* **2002**, *13*, 260–274.
16. Kwok, Y.K.; Ahmad, I. Benchmarking and Comparison of the Task Graph Scheduling Algorithms. *Journal of Parallel and Distributed Computing* **1999**, *59*, 381–422.
17. Li, K.; et al. Heuristics for Scheduling DAGs in Grid Computing. In Proceedings of the IEEE International Conference on High Performance Computing, 2004, pp. 1–8.
18. Casanova, H.; et al. Heuristics for Scheduling DAG Applications on Heterogeneous Systems. *Parallel and Distributed Computing* **2000**, *61*, 1337–1361.
19. Goldberg, D.E. *Genetic Algorithms in Search, Optimization, and Machine Learning*; Addison-Wesley, 1989.
20. Kennedy, J.; Eberhart, R. Particle Swarm Optimization. *Proceedings of IEEE International Conference on Neural Networks* **1995**, *4*, 1942–1948.

 doi:10.20944/preprints202601.1200.v1

21. Lozano, J.A.; et al. Towards a New Evolutionary Computation: Advances in Estimation of Distribution Algorithms. *Springer Science & Business Media* **2006**.
22. Zhang, Y.; et al. Energy-efficient Task Scheduling in Cloud–Edge Computing. *IEEE Access* **2020**, *8*, 190734–190747.
23. Satyanarayanan, M. The Emergence of Edge Computing. *Computer* **2017**, *50*, 30–39.
24. Bellendorf, J.; Mann, Z.A. Classification of Optimization Problems in Fog Computing. *Future Generation Computer Systems* **2020**, *107*, 158–176.