

Hybrid Resource Allocation Technique Based on Double Q Learning and Metaheuristic Algorithms in Optimizing the Allocation of Tasks in Cloud Computing

V. Mahalakshmi¹, S. Poornima²

Research Scholar, Department of Computer Science, Vel's Institute of Science,
Technology and Advanced Studies, Chennai, Tamil Nadu, India¹

Department of Computer Science, Vels Institute of Science Technology and Advanced Studies (VISTAS),
Chennai, Tamil Nadu, India²

Abstract—Cloud computing (CC) has grown as a core component of modern computing facilities, providing users with adaptable, on-demand services. In dynamic environments, the effective utilization of resources is significant for performance, cost-effectiveness, and user satisfaction. The optimization of resource allocation is important in CC due to the dynamic and complex nature of cloud infrastructure. This study suggests a hybrid resource allocation procedure that combines Double Q-learning (DQ), a reinforcement learning (RL) approach, with meta-heuristic algorithms, including the Whale Optimization Algorithm (WOA), and it is compared with Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Genetic Algorithms (GA) to optimize task allocation in CC settings. This combined system overcomes task allocation challenges such as usage of resources, energy consumption, and SLA compliance by integrating Q-learning's adaptive decision-making skills with meta-heuristic methods' global search efficiency. The proposed approach dynamically adapts optimum resource allocation guidelines while considering a variety of options for near-optimal task scheduling and resource assignment. The results of the experiment reveal that the hybrid strategy outperforms the PSO, ACO, and GA techniques in terms of makespan, reaction time, throughput, resource utilization of resources, cost, and rate of resource violation. This study provides an effective approach to handling complex resource allocation difficulties in a variety of dynamic cloud environments.

Keywords—Cloud computing; resource allocation; double Q-learning; meta-heuristic algorithms; Whale Optimization Algorithm (WOA); Particle Swarm Optimization (PSO); Ant Colony Optimization (ACO); Genetic Algorithms (GA)

I. INTRODUCTION

The rise of CC and services indicated the beginning of a new era of computing, giving numerous advantages, including expansion, flexibility, and affordability. Resource Allocation in CC is the process of distributing resources across different users and applications in the cloud. These resources include memory, computational power, storage, and network bandwidth [1]. The exponential rise of CC users has resulted in an extraordinary surge in demand for computing resources to perform cloud services. As a result, it is necessary to evaluate

currently available resources such as virtual machines (VMs), CPU, RAM, and storage before allocating cloud services. The objective of resource allocation in the cloud is normally to assign resources to tasks (or virtual machines) in such a way that specific objectives are optimized, such as minimizing makespan, energy consumption, and maximizing resource utilization, throughput, and so on.

A. Cloud Computing

CC is a technological paradigm that enables customers to pay as they go for computing resources (e.g., servers, storage, databases, networking, software, and analytics) via the internet ("the cloud"). The design eliminates the need to own and manage physical data centers or servers, resulting in increased adaptability, scaling, and cost-efficiency. It is a model that provides on-demand access to a common pool of configurable computing assets (such as networks, storage, servers, applications, and services). These resources can be deployed and released with little management effort or intervention from the service provider. Cloud service models include Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). IaaS offers virtualized computational assets such as virtual computers, storage, and networks. PaaS allows developers to construct, test, and launch apps without having to manage the structure that underlies them. Software as a service (SaaS) is delivered via the internet, generally through a web browser.

Resource allocation is a fundamental part of CC. It includes controlling and allocating the underlying virtual and physical resources in order to ensure efficient operation, fulfill user requests, and maintain service-level agreements. In CC, resource allocation focuses on optimizing resource distribution to improve efficiency and reduce costs while maintaining stability and scalability. It entails determining which resources to dedicate to specific activities, how much to give away, and for how long [2]. The basic concepts of resource allocation include the following.

Elasticity: The ability to gain resources in a flexible manner is seen as the fundamental reason for implementing a CC system. Elasticity primarily supports the ability to build and

shrink virtual resources dynamically in response to the needs of cloud users. Elastic resource allocation is critical for properly managing cloud resources based on customer demand and cloud service provider (CSP's) resources [3]. Elastic resources are programs and infrastructure that can be accessed on request when traffic or workloads increase. Cloud companies like AWS and Google Cloud use elastic resources as an enterprise model to bill consumers on demand, much like a utility bill. The cloud ecosystem is built on these supply and demand economic cycles. EC2 servers are an outstanding example of elastic resources. If a firm only needs two servers to host their website but experiences a holiday traffic increase, they may simply deploy more elastic resources by increasing the number of EC2 servers from two to four. Once the traffic decreases, they can reduce the server count to two.

Multi-tenancy: Multi-tenancy refers to having several tenants. This refers to a collection of users, resources, or services. The multi-tenant application was designed so that multiple users could access the same logic at the same time. A Software-as-a-Service (SaaS) provider can run numerous instances of the application and deliver web access to multiple customers, each with their own particular configuration. Such a model enables data isolation because each tenant's data is invisible to others.

QoS (Quality of Service): In CC, QoS is a metric used to assess the efficiency of a service, such as a network or cloud service. It is a method of evaluating the functionality of a service and ensuring that it fulfills the requirements. QoS measurements may include availability, reliability, and response time. The Cloud environment is a massive collection of virtually accessible resources that carry out millions of computing operations in real time for resource provisioning. Allocation and scheduling are two key components of providing for quality of service. This covers sophisticated modules including identifying work requirements, determining resource availability, making allocation decisions, and scheduling activities. In the current setting, it is challenging to manage cloud resources because service providers aim to deliver resources to consumers at productive cost and time[4].

Cost Efficiency: Inefficiencies in the cloud tend to manifest through over-provisioning and inefficient use of resources, which has the effect of increasing costs of operation. Common problems range from using too much CPU, storing too much data, and wasting on networking. If businesses overprovision their clouds, they pay for services not being utilized fully. Proper cloud cost optimization solutions are imperative in a bid to avoid such problems and enhance efficiency.

Workload Management: Strategic resource allocation, load balancing, and automation are required for effective cloud workload management to provide the highest achievable performance, efficiency, and cost savings in multiple cloud services and environments. Effective workload distribution in data centers enables business organizations to increase IT productivity, cut costs, and improve system effectiveness.

Dynamic resource allocation refers to the real-time assignment and reassignment of computing resources (e.g., CPU, memory, storage, and bandwidth) according to changing demands from users, software, or services in a cloud

infrastructure. This method guarantees optimal resource utilization, cost efficiency, and responsiveness to dynamic workloads. Resource provisioning was done employing VM technology. It is argued that using a virtualized environment will reduce average job response times in addition to performing tasks on the basis of available resources. Consequently, users are allocated virtual machines based on work criteria. Efficient and dynamic utilization of cloud resources could help distribute the load and prevent occurrences such as system performance degradation. Meta-heuristics are techniques used to solve issues with a broad domain space that are difficult to research using conventional methods.

They are widely used in CC for resource allocation because of their capacity to handle complicated, multi-objective, and dynamic optimization problems. These algorithms have grown dynamically in previous decades. Scholars have been encouraged to use meta-heuristic algorithms to solve NP-hard issues. The benefit of employing these types of algorithms is that they can detect the best solution with less processing effort and fewer iterations than simple heuristic methods. The meta-heuristic method has some benefits, including simplicity, flexibility, source-independent solutions, and the ability not to become trapped in local optimum situations [5]. For effective resource allocation on the cloud, certain meta-heuristic algorithms such as ACO [6], PSO [7], Artificial Bee Colony (ABC)[8], Rock Hyrax Optimization (RHO)[9], and Bacterial Foraging Optimization (BFO)[10] were used. Resource distribution optimization methods are used to efficiently make allocations while satisfying constraints such as cost, power, and performance, and establishing balance among conflicting goals is very important in CC. The research goals are as follows:

- To design and test a hybrid resource allocation model incorporating Double Q-learning (DQ) and meta-heuristic algorithms to schedule tasks and allocate resources in dynamic CC environments efficiently.
- To compare the designed hybrid approach with other meta-heuristic algorithms such as Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Genetic Algorithm (GA) in cloud computing environments.
- To attain optimal cloud performance metrics like makespan, response time, throughput, resource utilization, cost, and rate of resource violation.

The remaining part of this research is structured as follows. Section II discusses some existing works on resource allocation in cloud computing through reinforcement learning and meta-heuristics. Section III presents the proposed hybrid methodology by integrating Double Q-learning and the Whale Optimization Algorithm. Section IV discusses experimental findings and observations, and Section V concludes the work with key findings and future work directions.

II. LITERATURE REVIEW

In recent years, with the growing complexity of cloud resource scheduling requirements and the dynamic nature of cloud workloads, machine learning, deep learning,

reinforcement learning, and meta-heuristic optimization have played an increasing role in recent studies on cloud resource allocation. Their focus is on better use of resources, optimization of task scheduling, workload forecasting, load balancing, cost optimization, and service quality. Recently, there has been a lot of development going on in intelligent cloud resource management, which is summarized in the following studies.

Zheng et al. (2024) designed a prediction model by using XGBoost and LSTM to predict workload trends over time. The model is trained on historical data from an enormous cloud system with 1000 servers and more than 52 million data points. A dynamic resource scaling strategy is proposed with the use of the output of the prediction model and real-time system status inputs to make active allocation decisions [11]. Alizadeh et al. (2024) suggest an autonomous VM resource allocator based on the Clipped Double Deep Q-Learning (CDDQL) Algorithm and the meta-heuristic PSO. The tasks are prioritized through the use of the PSO algorithm, while CDDQL serves as the core of the autonomous system (Auto-CDDQL) in order to assign the proper VM resources to the tasks[12].

Zhang et al. (2024) present an action-constrained deep RL (DRL) technique for safely allocating computer resources and providing an optimal resource scheduling strategy. Task scheduling in dynamic cloud-based systems is a challenging task due to varying workload requirements, changing resource availability, and inconsistent task priorities. Traditional methods of optimization are generally inefficient in handling complex limitations [13]. To solve the above problems, Bajpai et al. 2024 introduce the Q-Whale algorithm (QWA) and the

SARSA-Whale algorithm (SWA), a novel hybrid method that combines the WOA and RL methods [14]. Khani et al. review recent works on resource distribution in 5G cloud-based radio access networks (C-RAN) using DRL algorithms. It explores the potential of DRL to learn complex making decisions policies without intervention [15].

Gogula et al., 2024 try to formulate a task handling framework with decision-making capability to choose the best resource in real time for managing different and complicated applications on Virtual machines [16]. It applies a modified PSO in order to fulfill the user's deadline. Gongada et al., 2024 suggest a new task scheduling optimization framework that integrates Fruit Fly Optimization (FFO) and Convolutional Neural Networks (CNN). It highlights the importance of efficient resource allocation and management in CC to meet increasing demands for computing resources at reduced costs and enhanced overall system performance [17].

Aditi et al., 2025 compare a number of DL models, i.e., Recurrent Neural Networks, Residual Networks (ResNets), and CNN, to optimize the utilization of cloud resources and system performance [18]. Muniswamy et al.,2024 present a joint optimization of LB and resource allocation in the cloud with the help of an optimal container management method. It aims to enhance scheduling efficiency and be cost-saving through upgrading the requested remotely by users' container schedule. An enhanced backtracking search optimization (IBSO) method is applied to allocate resources between end-users/IoT devices and the cloud subject to service-level agreements [19]. Table I lists the recent related studies discovered in the literature for resource allocation in CC.

TABLE I. RELATED STUDIES IN THE LITERATURE FOR RESOURCE ALLOCATION IN CC

Author /Year	Methods	Data set	Evaluation Criteria	Limitations
Zheng et al., 2024	XGBoost and LSTM	Historical and real time data	Resource Utilization, Resource Violation Rate, Energy Efficiency	It works well on weel crafted features, not applicable for real time , computationally expensive
Alizadeh et al., 2024	CDDQL	c-hilo	Makes span, response time, task completion, resource utilization, and energy consumption rate	Computational complexity, takes a long time to converge, overfits to specific patterns
Zhang et al., 2024	DRL	Simulated data	Cost, delay, and security	High computational cost, long training time, scalability issues
Gogula et al., 2024	PSO	User-generated data	Resource utilization	Premature convergence, depends heavily on inertia weight and learning factors
Gongada et al., 2024	FFO and CNN	Historical data	Resource utilization and average response time	Only suitable for image processing
Aditi et al., 2025	CNN	MNIST dataset	Accuracy	Only suitable for image processing
Muniswamy et al.,2024	IBSO	Simulated data	Energy consumption, number of active servers, number of interruptions, total cost, runtime, and statistical measures.	Not suitable for real-time; requires tuning

Existing studies on cloud resource allocation have several limitations. Most of the approaches, such as XGBoost, LSTM, and DRL, are computationally costly, take a long time to train, and are not suitable for real-time applications. Approaches such as PSO are prone to premature convergence and are highly sensitive to parameter settings, whereas CNN-based models are primarily restricted to image processing tasks and are not suitable for application in different cloud environments. Besides, certain approaches have a tendency to overfit to particular data patterns and are not as agile as needed to handle dynamic and unpredictable workloads of cloud computing.

In spite of significant progress, there is an evident research gap in creating adaptive, scalable, and efficient resource allocation techniques for adaptive cloud environments. Existing works do not properly compromise between decision accuracy and global search ability, causing inefficient task scheduling and resource handling. Relatively few pieces of work have dealt with combining reinforcement learning methods such as Double Q-learning with meta-heuristic optimization techniques to combine local decision-making and global optimality. In addition, scarce research critically analyzes such hybrid approaches against multiple performance

measures like makespan, cost, response time, and SLA fulfillment.

III. METHODOLOGY

This section describes the technique utilized to achieve the research objectives. The hybrid technique utilizes the adaptive decision-making capabilities of Q-learning and the global exploration efficiency of meta-heuristic algorithms. The framework shown in Fig. 1 combines the information of the cloud workload with adaptive decision-making (based on Double Q-Learning) and global exploration and local exploitation (WOA). For comparison, PSO, ACO and GA are some alternative optimization techniques. The framework takes care of resource needs, checks out the candidate allocations, and generates optimal resource allocations taking into account utilization, performance, QoS, and SLA restrictions.

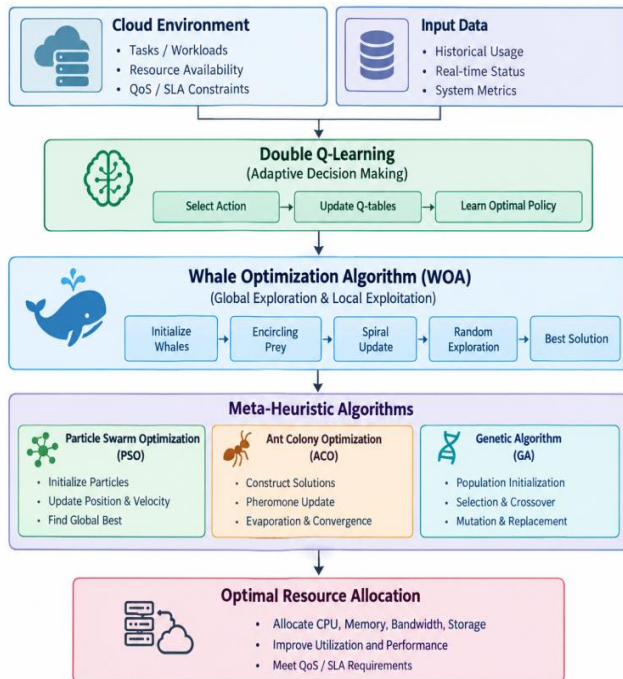


Fig. 1. Proposed hybrid double Q-learning and whale optimization algorithm framework.

A. Double Q-Learning

Q-learning is a RL method that teaches an agent how to perform optimal behaviors in a given environment in order to maximize cumulative rewards. It belongs to the value-based technique family and is especially beneficial for decision-making situations in which the agent communicates with the surrounding environment over time. The Bellman equation is used by the Q-learning algorithm to repeatedly update the Q-value for each state-action pair.

$$Q(s, a) \leftarrow q(s, a) + \alpha \{r + \max_{a'} (Q(s', a') - Q(s, a))\} \quad (1)$$

where, $Q(s, a)$: current estimate of the Q-value, α , the learning rate, r —immediate reward after the action a in state s , γ —discount factor[20].

An improved version of the initial Q-learning algorithm, double Q-learning was derived to solve the problem of

overestimation of the action value. In standard Q-learning, the value of an action is estimated using the maximum Q-value in the Q-table (or Q-function) for the subsequent state. As the action of maximum value is selected based on a noisy estimate instead of the actual value of the action, this could be a cause of overestimation. This bias of overestimation can cause inefficient policies or reduce learning speed. To overcome this bias, double Q-learning uses two separate Q-value estimators (Q_1 and Q_2). The steps in Double Q-learning are as follows.

Step 1: Two Q-value tables: Instead of maintaining a single Q-value table, Double Q-learning maintains two separate Q-value tables: $Q_1(s, a)$ and $Q_2(s, a)$. The two are updated independently.

Step 2: Action Selection: During learning, one of the Q-tables is used to select an action while learning.

Step 3: Update Process: While updating the Q-values, Double Q-learning employs the other Q-table to avoid the overestimation bias:

The action selected from the first Q-table (Q_1) is assessed according to the second Q-table (Q_2).

Particularly, the update of Q_1 and Q_2 is computed as follows:

$$Q_1(s, a) \leftarrow Q_1(s, a) + \alpha (r + \gamma Q_2(s', \arg \max_{a'} Q_1(s', a')) - Q_1(s, a)) \quad (2)$$

$$Q_2(s, a) \leftarrow Q_2(s, a) + \alpha (r + \gamma Q_1(s', \arg \max_{a'} Q_2(s', a')) - Q_2(s, a)) \quad (3)$$

This update employs the Q-table, which was not utilized to make the action choice, to calculate the next state's value and minimize overestimation of actions' values.

B. Whale Optimization Algorithm (WOA)

The WOA is a population-based optimization algorithm inspired by nature and based on humpback whale hunting behavior. The algorithm used belongs to the family of algorithms for swarm intelligence and was motivated by humpback whales' social behavior and hunting skills, especially their bubble-net feeding strategy. The WOA algorithmic Steps are:

- Randomly initialize a set of whales (solutions). Each whale carries an occupation in the solution space.
- Update Position: The algorithm updates the position of a whale by using three different strategies.
- Encircling Prey: Update the position of the whale using the distance between the current position of the whale and the best solution encountered so far (the prey).
- Spiral Update Position is inspired by the spiraling movement of whales while heading to the target. The update uses a spiral function in the equation.
- Random Exploration: During exploration, the algorithm uses random search to scan across the solution space.

- The above improvements make the algorithm mix global exploration (scanning widely across the solution space) with local exploitation (scanning narrowly around the current best so far).
- Objective Function Assessment: Each whale assesses the objective function (fitness function) so that it can identify how good the solution is. The fitness function is problem-specific.
- Update Best Solution: The algorithm holds the best solution obtained during the search and updates it.
- Termination: The algorithm updates the position of the whales at regular intervals and examines solutions until a stopping condition is met.

The flowchart for the hybrid Q-Whale approach is shown in Fig. 2.

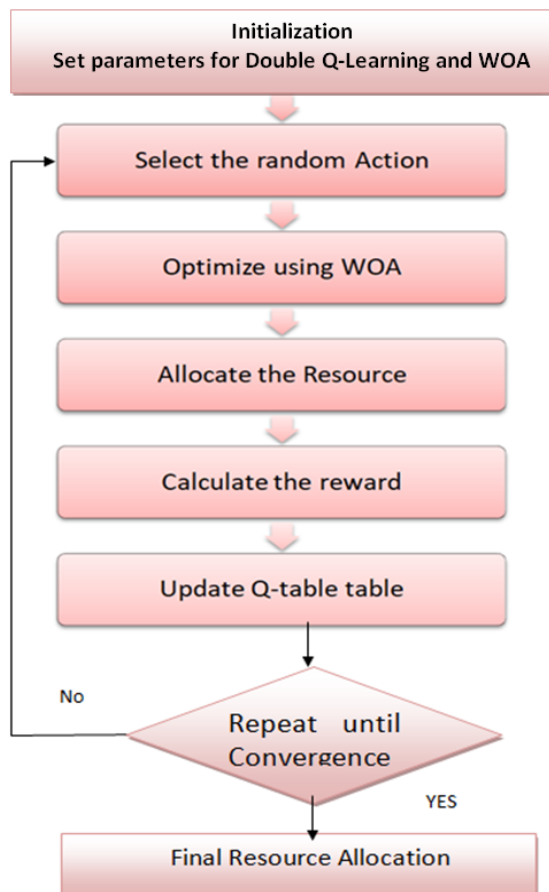


Fig. 2. Flowchart for the hybrid double Q-learning and WOA.

C. Particle Swarm Intelligence (PSO)

PSO is a school of fish or flocking-inspired algorithm that has been highly successful in addressing complicated optimization challenges, particularly in cloud computing resource allocation, communication networks, and energy consumption. In resource allocation, PSO can be utilized to allocate the resources (e.g., CPU capacity, memory, bandwidth, or storage) among the various tasks, users, or services in a way

that is most cost-effective, efficient, and satisfies Quality of Service (QoS) needs.

PSO consists of a collection of candidate solutions known as particles, and each particle is associated with one of the alternatives in the solution space. All the particles move their position in the solution space based on their local information and information of adjacent particles. The key steps in PSO are:

- Problem Description: Particle position indicates the possibility of resource allocation. For instance, an allocation can be manifested as a vector of resource allocation over many tasks or users. The objective function relates to how usable or optimal a given allocation is, e.g., least cost, energy consumption, or makespan not surpassing intermediate resource limits.
- The initialization step: Particles are initialized randomly in a solution space. Each particle is an instance of a unique distribution of resources.
- The fitness function calculates the performance metrics (e.g., cost, QoS, energy usage) and requirements (e.g., SLA, resource availability) of a solution of each particle. For example, in CC, the fitness function can optimize idle resources, load balance, or reduce energy consumption with respect to quality of service constraints.
- Location and Velocity Update: The position and velocity of each particle are updated based on the PSO equations, considering its individual best and global best solution. The algorithm encourages good search in the solution space by a balance of exploration (search for new areas) and exploitation (local improvement of the current good solutions).
- Convergence: Particle positions are updated at intervals and converge to the global optimal solution, and make the fullest use of the resources in the long run. The algorithm stops when repetitions are at maximum or when improvement in solution stops [21].

D. Ant Colony Optimization (ACO)

Ant Colony Optimization (ACO) is a nature-based optimization algorithm imitating ants' foraging behavior and, more significantly, how pheromone communication enables them to select the best path to food sources. ACO performs best on extremely large combinatorial optimization problems such as CC resource allocation, network optimization, scheduling jobs, and energy distribution. The general ACO method for resource allocation is as follows:

Initialization: The pheromone values are initialized on each edge (or decision variable) of the graph at the beginning. Ants are allocated to the graph in an effort to avoid experimenting with various schemes of resource allocation.

Solution Construction: One ant develops the solution by moving from a node (resource) to another, applying a probability decision rule on pheromone values and a heuristic function.

Pheromone Update: Once the solution is constructed, ants consider the quality of the solution with respect to an objective function (e.g., cost or performance). The concentration of pheromone on traveled paths is updated based on solution quality. Paths to better quality solutions have greater pheromone concentration and thus are more likely to be visited by following ants.

Evaporation: So that locally optimal solutions are avoided, the pheromone concentration decreases over time, and thus the impact of past routes is restricted.

Convergence: With more and more ants searching through different allocation paths, at some point the process converges to a phase where it uses an approximately optimal or optimal means of resource allocation.

ACO provides an efficient and universal solution to resource optimization in CC computer networks. Its capability of exploring diversified solutions and wide application in various situations renders it qualified for adaptive large-scale cloud systems with a dynamic nature. However, scalability, dynamic adaptability, and multi-objective optimization for practical use are all areas to be tackled [22].

E. Genetic Algorithms (GA)

Genetic Algorithms (GAs) are a powerful, nature-inspired optimization method based on genetics and natural selection. GAs have been used successfully to solve allocation problems in numerous fields, including cloud computing. Genetic algorithms are best suited to address complicated optimization problems, such as the resource allocation problem in cloud infrastructure, where the solution space is extremely huge, and objectives may be conflicting.

GAs operate on a population of possible solutions in which each solution is an individual member of the population. In cloud resource allocation, there can be various representations based on the given problem, which are: Chromosome Encoding: A GA individual or a chromosome might represent resource allocation task-specific. Each gene of the chromosome would be coded to some activity or resource, and resource allocations would be calculated based on the value of the gene (e.g., virtual machine to an activity, CPU or memory allocated, etc.). The standard Genetic Algorithm resource allocation process is the following:

Initialization: Random population of solutions (chromosomes) is created. Each solution contains one valid resource allocation.

Fitness Test: Each individual (resource allocation) is evaluated on the basis of a fitness function. The quality of the solution is defined by the fitness function. For example, it may penalize allocations that use more than available resources, excessive power consumption, or long runtimes.

Selection: Fittest individuals are selected on the basis of fitness degree. One may apply different selection strategies, for example:

- **Roulette Wheel Selection:** Probabilistically select individuals based on fitness.

- **Tournament Selection:** A method of choosing a pair at random from a set of people and selecting the winner.

Crossover (recombination) is an operation wherein two parent solutions are blended to produce offspring. In the case of resource allocation, this would be swapping portions of the chromosome (resource allocations) between the two individuals with the hope of creating new resource allocation solutions.

Mutation creates small random changes in offspring. It is done by assigning an asset at random to some other task or by assigning assets to replicate varying dynamics in the environment.

Replacement: The offspring are then inserted into the population, replacing the originals partially, usually based on their fitness.

Termination: The above process is run for a finite number of generations or until it achieves an optimal solution (i.e., population converges). Genetic algorithms are best suited to address resource allocation problems in cloud computing. They can search large spaces and optimize multi-objective problems with high levels, making them best suited to optimize resources in clouds. Scalability, variable environments, and convergence must be addressed prior to being implemented in actual clouds[23].

Algorithm: Hybrid Q-Whale Resource Allocation

Input: Cloud tasks, available resources, QoS/SLA constraints

Output: Optimized resource allocation

1. Initialize Q1, Q2 and the whale population.
 2. Evaluate the initial whale solutions.
 3. Identify the best whale solution.
 4. Repeat until convergence:
 - a. Observe the current cloud state.
 - b. Select an action using Double Q-Learning.
 - c. Execute the action and obtain the reward.
 - d. Update Q1 or Q2 using the other Q-table.
 - e. Update whale positions using WOA.
 - f. Evaluate candidate allocations.
 - g. Update the best solution.
 - h. Check resource and QoS/SLA constraints.
 5. Return the best resource allocation.
-

IV. RESULTS AND FINDINGS

This section discusses the tools and the parameters used to evaluate the resource allocation strategies. The proposed DQ-Whale resource allocation framework has been simulated and tested in the CloudSim simulation environment. A diverse cloud infrastructure was simulated using virtual machines with varying CPU sizes, memory amounts, and bandwidth and storage sizes. Dynamic workloads were generated by multiple cloud tasks that had different levels of computational requirements, priorities, and QoS constraints. The algorithms (DQ-Whale, PSO, ACO, and GA) were all implemented with the same workload and resources to facilitate a fair comparison. In simulation, task allocation and scheduling

decisions were continually analyzed depending on the makespan, throughput, response time, resource utilization, cost, and resource violation rate. The experiments were repeated with the same setup to evaluate the allocation efficiency and scheduling performance.

- Makespan: Total time needed to perform all tasks, and reducing makespan is a typical goal in resource allocation [24].
- Response time: The time it takes for a system to reply to a task request. Lower response times suggest more effective resource allocation [25].

Response time = completion time – task arrival time

- Throughput: Measures the number of jobs accomplished per unit of time. High throughput suggests efficient resource allocation [26].
- Resource Utilization: It tracks how effectively cloud resources (such as CPU, memory, and storage) are used. It represents the percentage of the resources accessible that are actively utilized. High utilization indicates efficient resource deployment but may result in overburden [27].

$$\text{Utilization rate} = \frac{\text{Resource used}}{\text{Total Resource capacity}} * 100\%$$

- Cost: The total cost taken for task allocation; lower values indicate cost effectiveness
- Resource violation rate: The percentage of time resources are overutilized or underutilized. A lower rate indicates better performance [28].

For more reliability in the experimental evaluation process, five different simulation runs of CloudSim were performed in each of the experiments with identical workload and resources of the five algorithms. All the same task characteristics, resources limitations and evaluation conditions were used for the DQ-Whale, PSO, ACO and GA to make a consistent comparison. The results are presented as mean $\pm$ standard deviation ($n = 5$ trials), and thus give information not only about the average performance in the trials, but also about the variation between the trials. Evaluated metrics: Makespan, throughput, response time, resource utilization, allocation cost, and resource violation rate. These results are therefore only

observations of the algorithms based on a specific set of simulation conditions and should not be considered as statistically significant proof of the superiority of the algorithms.

Table II lists the performance measure values for the proposed Q-Whale, PSO, ACO, and GA algorithms. The makespan time for Q-Whale is 20 which is found to be low compared to all the other methods, indicating better performance in the hybrid Q-Whale algorithm. A value of 50 tasks/mts shows that a higher value in throughput indicates that the number of tasks completed per minute is higher in Q-Whale. The average time taken to respond to the task is low in the case of Q-whale. Q-Whale is cost-effective, and the resource violation rate shows only 2%, which is higher for Q-Whale compared to the contrasted methods PSO, ACO, and GA.

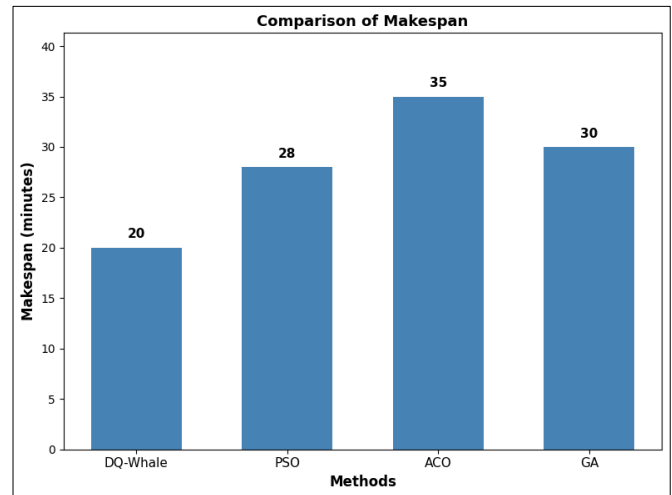


Fig. 3. Methods vs. makespan.

Fig. 3 shows a shorter makespan time of 20 minutes for the hybrid Q-Whale method and a maximum of 35 minutes for ACO. The nature of makespan time in the allocation of resources refers to how the overall completion time of a group of tasks (makespan) responds to or is influenced by various variables and methods in a resource allocation system. It is an important performance metric in CC and distributed systems since it has a direct impact on resource efficiency and system productivity.

TABLE II. PERFORMANCE MEASURES FOR RESOURCE ALLOCATION STRATEGIES

Methods/ Measures	Make span (mts)	Throughput (tasks/mts)	Response time (seconds/tasks)	Resource utilization (%)	Cost (units)	Resource violation Rate (%)
DQ-Whale (mts)	20	50	5	95	1000	2
PSO	28	45	8	88	1200	5
ACO	35	40	10	85	1500	7
GA	30	42	7	90	1300	4

Fig. 4 shows higher throughput for DQ-whale than for the PSO, ACO and GA methods. Throughput in resource allocation refers to how effectively a system can handle jobs over time, and it is an important performance indicator in CC,

distributed systems, and task scheduling. Throughput influences a system's overall productivity and capacity to satisfy performance objectives.

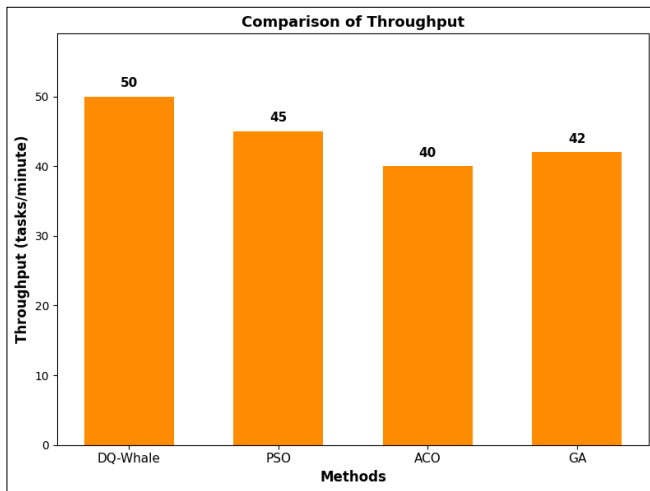


Fig. 4. Methods vs throughput.

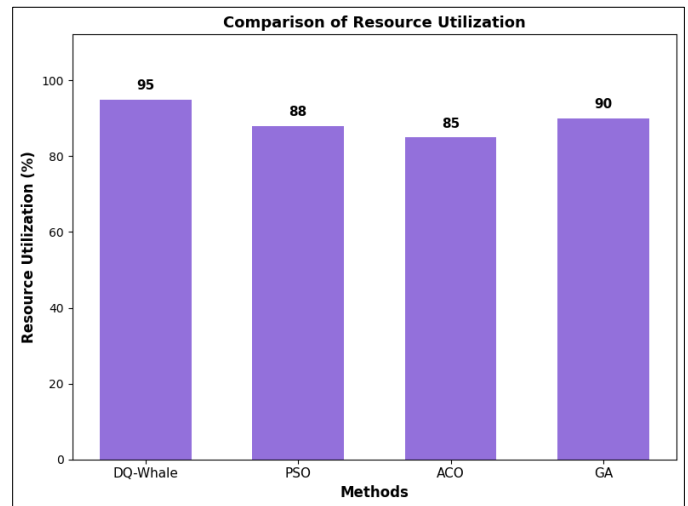


Fig. 6. Methods vs. resource utilization.

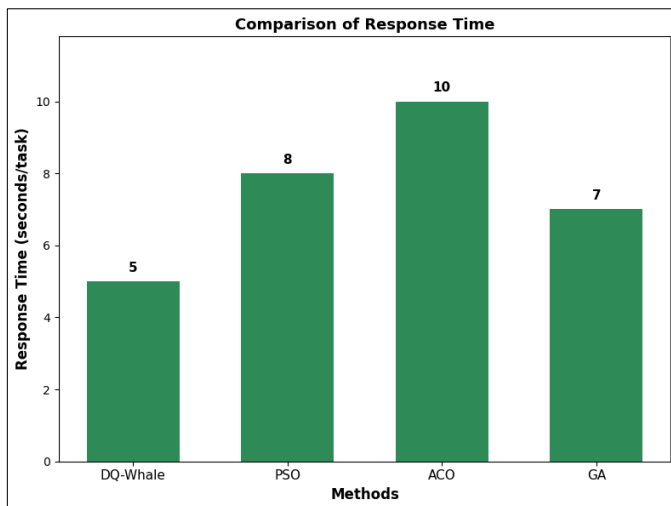


Fig. 5. Methods vs. response time.

Fig. 5 presents the response time for the proposed DQ-Whale method and other contrasted methods used for evaluation. The time taken to get a reply for tasks is low for the DQ-Whale method. The concept of reaction time in resource allocation relates to how soon a system can reply to and process a request or task once it has been received. It is an important performance indicator in computers, cloud services, and distributed systems since it impacts both user experience and system efficiency. Techniques that are adaptive to current network conditions, distribute load equitably, and properly prioritize jobs can assist in reducing reaction time.

As shown in Fig. 6, the resource utilization for DQ-Whale shows 95%, PSO as 88%, ACO as 85% and GA as 90% and is found to be high for hybrid DQ-Whale methods than the other methods. Resource utilization in resource allocation describes how efficiently and effectively the assets of a system (such as CPU, memory, bandwidth, and so on) are used to complete tasks. Proper resource use is critical for increasing performance, lowering expenses for operations, and ensuring that systems run efficiently.

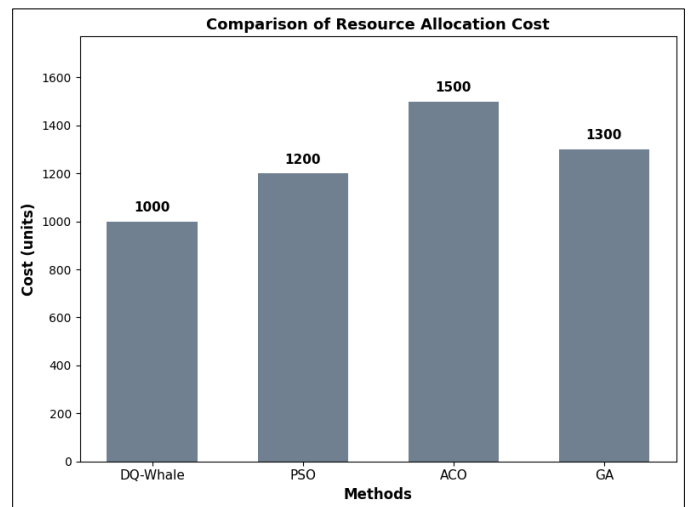


Fig. 7. Methods vs. cost.

Fig. 7 shows the decreased cost in units for proposed Q-whale method than the PSO, ACO and GA techniques. The cost in resource allocation relates to the numerous expenditures paid in order to properly use and manage computer resources inside a system. Cost concerns are critical in resource allocation because they impact decisions about infrastructure, optimizing performance, and overall system efficiency.

Fig. 8 shows a lower resource violation rate for DQ-Whale algorithm than for the PSO, ACO and GA strategies. In resource allocation, the resource violation rate refers to the frequency or possibility of resource consumption exceeding the allotted limitations or restrictions for a job or system. Resource violations can signal inefficiencies, future system overloads, or policy breaches; therefore, knowing this statistic is critical for improving resource allocation techniques. Q-Whale, PSO, ACO, and GA are examples of algorithms that aim to maximize throughput by efficiently distributing jobs to available resources. Their efficacy is determined by their ability to adapt to changing workloads, investigate the solution space, and strike a balance between exploration and exploitation.

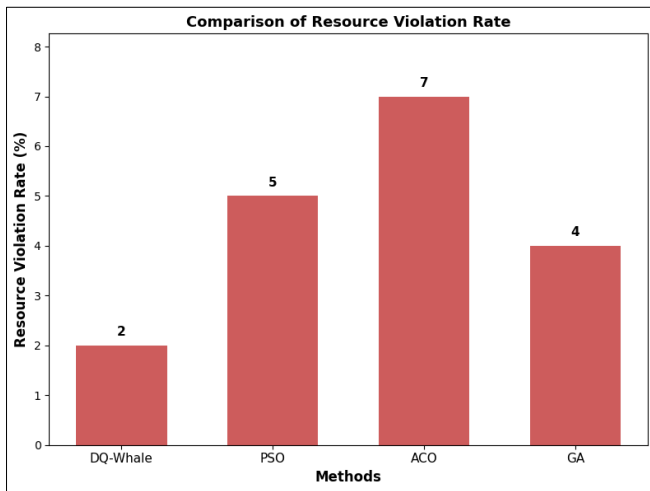


Fig. 8. Methods vs. resource violation rate.

Fig. 9 is a comparison of DQ-Whale, PSO, ACO and GA in terms of six measures of cloud resource allocation. The makespan, throughput, latency, resource utilization, cost, and resource violation rate are good properties of DQ-Whale. It is adaptive in decision-making, which enables faster allocation and efficient utilization of resources with varying workloads. PSOs offer responsive allocation and can have local optima issues. ACO has moderate performance and convergence rate, especially in complex tasks. Good resource allocation is provided by GA, but it may need more iterations based on the complexity of the workload and population size. In general, the qualitative comparison shows the various features of the methods evaluated and the wide applicability of DQ-Whale for cloud resource management.

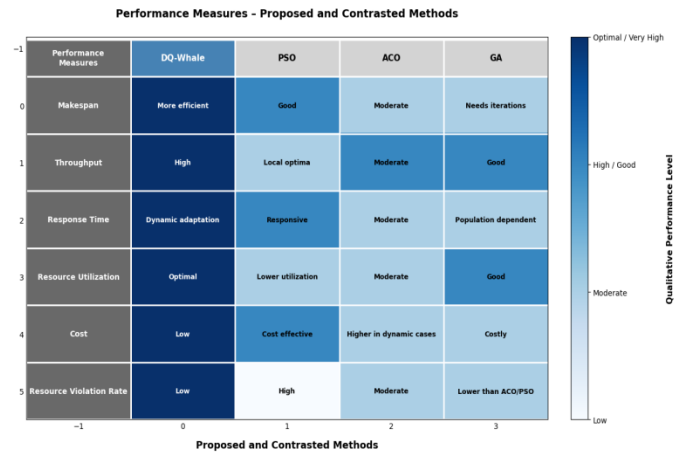


Fig. 9. Qualitative performance comparison of DQ-Whale, PSO, ACO, and GA across cloud resource allocation measures.

Table III points out that the main difference in methodology of DQ-Whale is the combination of Double Q-Learning with WOA, not the use of WOA. The reinforcement learning-WOA algorithms are also used as Q-Whale and SARSA-Whale, but with standard Q-Learning and SARSA, respectively. The innovation of DQ-Whale is that it employs two value estimators to diminish the potential overestimation in the process of action-value learning and keeps the WOA-based exploration and exploitation. The table also highlights that there is no numerical superiority of these methods over Q-Whale or SARSA-Whale for the experiments since the latter two methods were not tested with the same CloudSim configuration as described in the present experiments.

TABLE III. COMPARATIVE CHARACTERISTICS OF REINFORCEMENT-LEARNING-BASED HYBRID RESOURCE ALLOCATION APPROACHES

Performance/Methodological Aspect	DQ-Whale (Proposed)	Q-Whale	SARSA-Whale
Reinforcement learning strategy	Double Q-Learning	Q-Learning	SARSA
Value estimation	Two Q-tables	Single Q-table	Single action-value function
Action policy	Adaptive action selection using two value estimates	Q-value-based action selection	On-policy action selection
Overestimation control	Explicitly reduces Q-value overestimation	More susceptible to Q-value overestimation	Does not specifically employ Double Q-learning correction
Optimization mechanism	Whale Optimization Algorithm (WOA)	Whale Optimization Algorithm (WOA)	Whale Optimization Algorithm (WOA)
Exploration	Q-learning exploration combined with WOA exploration	Q-learning exploration combined with WOA	SARSA policy exploration combined with WOA
Exploitation	Double Q-value guidance with WOA exploitation	Q-value guidance with WOA exploitation	SARSA-based policy with WOA exploitation
Adaptation to changing states	High, through double-estimator learning	Adaptive through Q-value updates	Adaptive through on-policy updates
Allocation decision mechanism	State-action learning with reduced value-estimation bias	State-action learning	State-action-next-action learning
Main methodological characteristic	Combines bias-reduced reinforcement learning with population-based optimization	Combines conventional Q-learning with WOA	Combines on-policy SARSA learning with WOA
Expected computational consideration	Additional Q-table maintenance	Lower Q-learning memory requirement	Additional policy-transition evaluation
Direct comparability with present study	Evaluated under the proposed CloudSim configuration	Requires identical reimplementations and experimental conditions	Requires identical reimplementations and experimental conditions

There are still several issues in the current methods of allocation of cloud resources that can be improved in the future. A variety of studies have been conducted on algorithms

in simulated and relatively controlled workloads, and this may not accurately reflect workloads in a highly dynamic cloud environment. Other approaches can have a tendency to

converge prematurely, suffer from local optima, have a higher computational load, and lack scalability with the number of tasks and resources. Moreover, few works have focused on multi-objective optimization that considers the trade-offs among the cost, energy consumption, latency, resource utilization, and QoS at the same time. Following researchers can consider adaptive hybrid algorithms with real world cloud workloads, online learning, multi objective optimization and distributed decision making in the future. Performance in edge-cloud and heterogeneous environments, as well as in case of workload uncertainty, failures of resources, and varying service level requirements, should be further studied.

V. CONCLUSION

The proposed Double Q-Learning–Whale Optimization (DQ-Whale) framework is an adaptive framework to allocate cloud resources using the decision-making process of the Double Q-Learning algorithm along with the exploration and exploitation ability of the Whale Optimization Algorithm (WOA). The experimental outcomes shown with CloudSim demonstrate good performance when compared with PSO, ACO, and GA for the simulation conditions with respect to the metrics, such as makespan, throughput, response time, resource utilization, cost and resource violation rate. But there are some drawbacks to this approach, DQ-Whale. Two Q-tables and a number of whale solutions increase the computational and memory cost; its performance might be dependent on the WOA parameters like the number of the whale population, the number of iterations and exploration–exploitation coefficients. Furthermore, the current assessment is based on a defined workload scale and simulated cloud environment, and thus the scalability, i.e., the ability to scale up to much larger workloads and significantly more heterogeneous infrastructures, has yet to be proven. In future, the numbers of tasks and virtual machines will be gradually increased to study scalability, parameter sensitivity, and ablation studies for the WOA, and computational overhead will be measured. Further experiments will investigate robustness against dynamic workloads and failures of resources, as well as changing QoS/SLA requirements. Repeated trials will also be performed together with the appropriate statistical significance tests for extensions of the multi-objective incorporating the energy consumption, cost, latency, and QoS.

REFERENCES

- [1] Mohammad, A., & Abbas, Y. (2024). Key Challenges of Cloud Computing Resource Allocation in Small and Medium Enterprises. *Digital*, 4(2), 372-388. <https://doi.org/10.3390/digital4020018>
- [2] Alam, Shahzeb, Muhammad Atif Ramzan, Muhammad Zubair, Ubaid Ullah, Rab Nawaz, and Rahmat Ullah. "AI-driven resource allocation in cloud computing: a systematic review revealing critical sustainability and evaluation gaps." *Computing* 108, no. 5 (2026): 67.
- [3] Thapa, Mitali, Kamal Kant Hiran, Simran Ajit Kumar Mohanty, Deepak Parashar, and Ankur Goyal. "Elastic Cloud Scaling for Small Models: Dynamic Resource Allocation as an Alternative to Training Large LLMs." *IEEE Access* (2026).
- [4] Vali, Ali Akbar, Sadoon Azizi, Mohammad Shojafar, and Rajkumar Buyya. "Energy-efficient resource management in microservices-based fog and edge computing: State-of-the-art and future directions." *ACM Computing Surveys* 58, no. 12 (2026): 1-42.
- [5] S. Singhal, N. Gupta, P. Berwal, Q. N. Naveed, A. Lasisi and A. W. Wodajo, "Energy Efficient Resource Allocation in Cloud Environment Using Metaheuristic Algorithm," in *IEEE Access*, vol. 11, pp. 126135-126146, 2023, doi: 10.1109/ACCESS.2023.3330434.
- [6] Lilhore, Umesh Kumar, Sarita Simaiya, KBV Brahma Rao, VVR Maheswara Rao, Yogesh Kumar Sharma, Roobaea Alroobaea, Hamed Alsufyani, Majed Alsafyani, and MD Monish Khan. "Hybrid DRL-enhanced ACO-WWO for efficient resource allocation and load-balancing in cloud computing." *International Journal of Computational Intelligence Systems* 18, no. 1 (2025): 148.
- [7] Nabi, Said, Masroor Ahmad, Muhammad Ibrahim, and Habib Hamam. "AdPSO: adaptive PSO-based task scheduling approach for cloud computing." *Sensors* 22, no. 3 (2022): 920.
- [8] Kruekaew, Boonhatai, and Warangkhan Kimpan. "Multi-objective task scheduling optimization for load balancing in cloud computing environment using hybrid artificial bee colony algorithm with reinforcement learning." *Ieee Access* 10 (2022): 17803-17818.
- [9] Singhal, Saurabh, Ashish Sharma, Pawan Kumar Verma, Mohit Kumar, Sahil Verma, Maninder Kaur, Joel JPC Rodrigues, Ruba Abu Khurma, and Maribel Garcia-Arenas. "Energy efficient load balancing algorithm for cloud computing using rock hyrax optimization." *IEEE access* 12 (2024): 48737-48749.
- [10] Hu, Yanjuan, Leiting Pan, Wenjun Lv, and Zhanli Wang. "Optimal selection of cloud manufacturing resources based on bacteria foraging optimization." *International Journal of Computer Integrated Manufacturing* 37, no. 1-2 (2024): 165-182.
- [11] Zheng H, Xu K, Zhang M, Tan H, Li H. Efficient resource allocation in cloud computing environments using AI-driven predictive analytics. *Applied and Computational Engineering*. 2024 Sep 10;82:17-23.
- [12] Alizadeh Javaheri SD, Ghaemi R, Monshizadeh Naeen H. An autonomous architecture based on reinforcement deep neural network for resource allocation in cloud computing. *Computing*. 2024 Feb;106(2):371-403.
- [13] Zhang H, Wang J, Zhang H, Bu C. Security computing resource allocation based on deep reinforcement learning in serverless multi-cloud edge computing. *Future Generation Computer Systems*. 2024 Feb 1;151:152-61.
- [14] Bajpai G, Singh P, Agarwal AK. Performance Enhancement of Cloud Resource Management by Using Optimized Reinforcement Learning Approaches. *Journal of Computational Analysis and Applications (JoCAAA)*. 2024 May 20;33(05):977-91.
- [15] Khani M, Jamali S, Sohrabi MK, Sadr MM, Ghaffari A. Resource allocation in 5G cloud-RAN using deep reinforcement learning algorithms: A review. *Transactions on Emerging Telecommunications Technologies*. 2024 Jan;35(1):e4929.
- [16] Gogula S, Sridhar P, Arvind S, Sethy A, Ragavendiran SP, Balasubramani P, Gupta KG. Optimized resource allocation in cloud computing for enhanced performance with modified particle swarm optimization. In *MATEC Web of Conferences 2024 (Vol. 392, p. 01140)*. EDP Sciences
- [17] Gongada TN, Desale GB, Ghodake SP, Sridharan K, Rao VS, El-Ebiary YA. Optimizing Resource Allocation in Cloud Environments using Fruit Fly Optimization and Convolutional Neural Networks. *International Journal of Advanced Computer Science & Applications*. 2024 May 1;15(5).
- [18] Aditi, Prasad VK, Gerogiannis VC, Kanavos A, Dansana D, Acharya B. Utilizing convolutional neural networks for resource allocation bottleneck analysis in cloud ecosystems. *Cluster Computing*. 2025 Feb;28(1):22.
- [19] Muniswamy S, Vignesh R. Joint optimization of load balancing and resource allocation in cloud environment using optimal container management strategy. *Concurrency and Computation: Practice and Experience*. 2024 May 30;36(12):e8035
- [20] Uma, J., Vivekanandan, P., & Shankar, S. (2022). Optimized intellectual resource scheduling using deep reinforcement Q-learning in cloud computing. *Transactions on Emerging Telecommunications Technologies*, 33(5), e4463.
- [21] Bahrepour, D., Evaznia, N., & Khodabakhshi, T. (2024). A New Resource Allocation Method Based on PSO in Cloud Computing. *International Journal of Web Research*, 7(2), 13-21.

- [22] Yin, C., Fang, Q., Li, H., Peng, Y., Xu, X., & Tang, D. (2024). An optimized resource scheduling algorithm based on GA and ACO algorithm in fog computing. *The Journal of Supercomputing*, 80(3), 4248-4285.
- [23] Behera, I., & Sobhanayak, S. (2024). Task scheduling optimization in heterogeneous cloud computing environments: A hybrid GA-GWO approach. *Journal of Parallel and Distributed Computing*, 183, 104766.
- [24] K. Karim, F., Ghorashi, S., Alkhalaf, S., HA Hamza, S., Ben Ishak, A., & Abdel-Khalek, S. (2024). Optimizing makespan and resource utilization in cloud computing environment via evolutionary scheduling approach. *PloS one*, 19(11), e0311814.
- [25] Alizadeh Javaheri, S. D., Ghaemi, R., & Monshizadeh Naeen, H. (2024). An autonomous architecture based on reinforcement deep neural network for resource allocation in cloud computing. *Computing*, 106(2), 371-403.
- [26] Jeyaraman, J., Bayani, S. V., & Malaiyappan, J. N. A. (2024). Optimizing Resource Allocation in Cloud Computing Using Machine Learning. *European Journal of Technology*, 8(3), 12-22.
- [27] Gong Y, Huang J, Liu B, Xu J, Wu B, Zhang Y. Dynamic resource allocation for virtual machine migration optimization using machine learning. *arXiv preprint arXiv:2403.13619*. 2024 Mar 20.
- [28] Mukherjee, A., De, D., & Buyya, R. (2024). Cloud Computing Resource Management. In *Resource Management in Distributed Systems* (pp. 17-37). Singapore: Springer Nature Singapore.