EUROPEAN JOURNAL OF PURE AND APPLIED MATHEMATICS
2025, Vol. 18, Issue 3, Article Number 6487
ISSN 1307-5543 – ejpam.com
Published by New York Business Global

Mathematical Modeling and Genetic Algorithm-Based Hyperheuristic Optimization for Quality of Service and Load Balancing in Cloud Communication Networks

Kassem Danach[1,], Wael Hosny Fouad Aly[2*], Samir Haddad[3]

[1] *Basic and Applied Sciences Research Center, Al Maaref University, Beirut, Lebanon*
[2] *College of Engineering and Technology, American University of the Middle East, Kuwait*
[3] *Department of Computer Science and Mathematics, Faculty of Arts and Sciences, University of Balamand, Koura 100, Lebanon*

Abstract. Ensuring Quality of Service (QoS) and efficient load balancing in cloud communication networks is critical for optimizing resource allocation, minimizing latency, and enhancing service reliability. Traditional load balancing strategies often fail to scale and adapt to dynamic cloud environments, resulting in network congestion, resource underutilization, and increased operational costs. This study presents a novel Genetic Algorithm (GA)-based hyperheuristic optimization framework integrated with a mathematical model for QoS-aware load balancing, designed to address the challenges of scalability and efficiency. The model is referred to as GAHO_{QoS}. We introduce valid inequalities to strengthen the optimization formulation, accelerating convergence and improving solution quality. Our GA-hyperheuristic framework dynamically selects and combines multiple low-level heuristics to optimize task allocation across cloud servers while adhering to QoS constraints such as latency, throughput, and energy efficiency. Experimental evaluations on a range of cloud communication scenarios demonstrate that GAHO_{QoS} significantly reduces service latency, balances workload distribution, and optimizes resource utilization. Comparative analysis with existing metaheuristic methods, including $GA-PSO$ and $SA-GA$, confirms that the proposed framework outperforms traditional approaches in terms of computational efficiency, scalability, and QoS satisfaction. GAHO_{QoS} provides an adaptable, computationally efficient solution for enhancing cloud network performance, contributing to the development of high-performance, energy-efficient, and robust cloud infrastructures.

Key Words and Phrases: Quality of Service (QoS), Load Balancing, Cloud Communication Networks, Genetic Algorithm (GA), Hyperheuristics, Resource Allocation, Valid Inequalities

*Corresponding author.
DOI: https://doi.org/10.29020/nybg.ejpam.v18i3.6487

Email addresses: kassem.danach@mu.edu.lb (K. Danach),
wael.aly@aum.edu.kw (W. H. F. Aly), samir.haddad@balamand.edu.lb (S. Haddad)

1. Introduction

Cloud communication networks have become essential infrastructures for modern computing, supporting a diverse range of services such as data storage, real-time applications, and high-performance computing [1]. As cloud systems evolve, ensuring *Quality of Service (QoS)* while maintaining efficient load balancing has become an increasingly complex challenge. These networks must dynamically allocate computing power, bandwidth, and storage resources to meet fluctuating user demands and service-level agreements [2]. Failures in load balancing can result in significant consequences, including increased latency, network congestion, inefficient resource utilization, and degraded service quality, all of which are critical concerns for cloud service providers [3].

Traditional load balancing methods, such as static allocation and round-robin scheduling, often fail to effectively optimize resource distribution, particularly in large-scale, multi-server cloud environments [4]. These approaches lack the adaptability necessary for managing dynamic workloads and the scalability required for cloud networks with rapidly growing user demands. Furthermore, existing optimization models face significant challenges in managing *multi-objective constraints*, such as minimizing latency, maximizing throughput, and improving energy efficiency, which are essential for ensuring the QoS in cloud environments [5]. In light of these issues, metaheuristic optimization techniques, particularly *Genetic Algorithms (GA)* and *hyperheuristics*, have gained attention as promising solutions to enhance load balancing performance in cloud networks [6].

This study introduces a *novel mathematical optimization model* for *QoS-aware load balancing* in cloud communication networks, integrating *valid inequalities* and a *GA-based hyperheuristic framework*. The proposed model aims to minimize service latency, optimize resource allocation, and improve overall network efficiency by leveraging *combinatorial optimization techniques* and *intelligent heuristic selection*. By dynamically selecting and combining multiple low-level heuristics, this framework enhances the scalability and adaptability of load balancing in the face of diverse network conditions and workloads. Cloud communication networks are often plagued by *load imbalances*, which lead to high latency, poor resource utilization, and degraded service quality. Traditional load balancing mechanisms lack the necessary flexibility and scalability to meet *QoS constraints* such as *throughput*, *reliability*, and *energy efficiency*. Existing mathematical models and metaheuristic approaches also struggle to handle the complex *multi-objective optimization challenges* inherent in cloud environments. Thus, there is an urgent need for an effective and scalable solution that enhances QoS while ensuring efficient load balancing across cloud resources.

The goal of this research is to develop a comprehensive solution to address the challenges in *QoS-aware load balancing* for cloud communication networks. To this end, the study develops a *mathematical optimization model* that considers QoS requirements, incorporates *valid inequalities* to improve mathematical formulation and computational efficiency, and implements a *GA-based hyperheuristic framework* that dynamically optimizes resource allocation. The proposed approach is evaluated against existing methods in terms of *latency*, *resource utilization*, and *solution quality*. This study further investigates the ef-

fectiveness of the mathematical model in enhancing QoS-aware load balancing, the role of valid inequalities in improving efficiency and solution quality, the comparative performance and scalability of the GAHO_{QoS} framework over traditional techniques, and the trade-offs between computational cost, network performance, and scalability in the proposed model.

Null Hypothesis (H_0): The proposed mathematical model of GAHO_{QoS} does not significantly improve QoS or *load balancing* compared to traditional methods. Alternative Hypothesis (H_1): The proposed mathematical model of GAHO_{QoS} significantly enhances QoS and load balancing, reducing latency and improving resource utilization in cloud communication networks. This research aims to offer a robust and scalable solution for QoS-aware load balancing in cloud networks by integrating combinatorial optimization, valid inequalities, and AI-driven heuristics. The findings from this study will contribute to the development of next-generation cloud infrastructures, optimizing service reliability, latency, and network efficiency.

The remainder of this paper is organized as follows. Section 2 provides a review of related work on load balancing strategies, QoS optimization, and hyperheuristic approaches in cloud computing. Section 3 discusses the reference models. Section 4 details the proposed GAHO_{QoS} framework, explaining its key components and algorithmic design. Section 5 describes the simulation and performance evaluation. Section 6 concludes the paper by summarizing the findings and outlining opportunities for further work.

2. Related Work

This section provides a comprehensive review of existing research efforts related to load balancing, Quality of Service (QoS) optimization, and heuristic-based scheduling within cloud communication networks. It examines various strategies and algorithms proposed in the literature to enhance resource allocation efficiency, minimize latency, and ensure service reliability. The review also highlights the strengths and limitations of current methodologies, setting the foundation for identifying research gaps and motivating the development of improved, intelligent scheduling frameworks.

Load balancing is a critical and ongoing challenge in cloud computing, where the objective is to dynamically and efficiently distribute incoming workloads across a pool of available servers and computational resources. This process is essential for maintaining the overall performance, reliability, and responsiveness of cloud-based systems, especially under varying demand conditions [1]. An effective load balancing strategy not only ensures that no single server is overwhelmed with excessive tasks but also contributes to optimized resource utilization, reduced task completion times, and improved user experience. Moreover, it helps in mitigating the risk of system bottlenecks, which can degrade service quality and potentially lead to system failures. Numerous approaches have been proposed to address this challenge, including static, dynamic, and hybrid strategies, each with distinct mechanisms for task allocation and decision-making [7–9].

2.1. Traditional Load Balancing Techniques

Traditional load balancing methods can be categorized into static and dynamic techniques. Static methods assign workloads to servers based on predefined rules and do not consider real-time changes in system load [4]. Common static approaches include:

- **Round-Robin:** Assigns tasks sequentially to each server in a cyclic manner, assuming equal processing power across nodes [10].
- **Weighted Round-Robin:** Extends Round-Robin by assigning weights to servers based on their computational capacity [2].
- **Least-Connection:** Assigns tasks to the server with the fewest active connections, ensuring that workloads are dynamically balanced [11].

Despite their simplicity and ease of implementation, static load balancing methods are inherently limited in their ability to adapt to changing system conditions. These methods rely on pre-defined rules or fixed workload distributions that do not consider real-time fluctuations in demand or server availability. As a result, static approaches often lead to inefficient resource allocation, where some servers remain underutilized while others become overloaded, especially in high-traffic or dynamic environments. This imbalance can significantly degrade system performance, increase response times, and reduce the overall reliability and scalability of cloud services.

2.2. Dynamic Load Balancing Techniques

Dynamic load balancing approaches continuously monitor system performance and reallocate workloads in real-time [12]. These methods offer enhanced adaptability to varying workloads, making them well-suited for dynamic cloud environments where demand can fluctuate unpredictably. However, this adaptability often comes at the cost of increased computational complexity and overhead due to the need for constant monitoring and decision-making. Among the prominent dynamic strategies is the Throttled Load Balancing method, which assigns incoming requests based on current server availability and actively rejects new requests if no suitable server is available to handle them [13]. Another effective approach is Active Monitoring Load Balancing, which continuously tracks the load on each server and redistributes tasks as needed to prevent overload and ensure optimal utilization [14]. In recent years, bio-inspired algorithms have shown considerable promise in dynamic load balancing. The Honeybee Foraging Algorithm (HFA), inspired by the natural foraging behavior of honeybees, dynamically allocates tasks based on the availability.

In recent years, ensuring Quality of Service (QoS) and achieving efficient load balancing in cloud communication networks has garnered significant attention due to the increasing complexity and scale of modern infrastructures. Several studies have proposed intelligent mechanisms to optimize resource allocation and service reliability. For instance, Aly et al. [15, 16] explored dynamic feedback and machine learning-based techniques for SDN,

highlighting the importance of adaptive strategies in network management. Additionally, Al-Tarawneh et al. [17] introduced a multi-criteria decision-making framework for trust-aware task offloading across heterogeneous environments, emphasizing fairness and system heterogeneity—key factors in achieving robust QoS. The relevance of dynamic and distributed strategies is also reflected in low-complexity differential schemes for wireless relay networks proposed by Alabed et al. [18], which align with the goals of reduced latency and energy efficiency. Moreover, the integration of agent-based models for risk analysis [19, 20] and UAV-based planning for network service continuity during crises [21] underscore the growing trend toward resilient, context-aware optimization techniques. Unlike these prior works, the proposed GAHO_{QoS} framework leverages a GA-based hyperheuristic approach to dynamically combine low-level heuristics and enhance the scalability, adaptability, and efficiency of load balancing under strict QoS constraints, outperforming traditional metaheuristics such as GA–PSO and SA–GA in complex cloud scenarios.

2.3. Metaheuristic-Based Load Balancing

Due to the inherent limitations of traditional optimization methods—such as gradient-based techniques, exhaustive search, or rule-based algorithms—in handling complex, high-dimensional, and non-convex problem spaces, researchers have increasingly turned their attention to metaheuristic-based approaches as a promising alternative for achieving improved scalability and performance [6]. A *metaheuristic* is a high-level, problem-independent optimization framework that orchestrates the behavior of subordinate heuristics to effectively explore and exploit large and often intractable search spaces. Unlike classical methods that may become trapped in local optima or require extensive problem-specific tuning, metaheuristics are designed to offer a more robust and adaptable mechanism for discovering global or near-global solutions.

Popular metaheuristic algorithms include, but are not limited to, Genetic Algorithms (GAs), Simulated Annealing (SA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO). These methods utilize a variety of biologically or physically inspired metaphors to balance the fundamental trade-off between *exploration* (searching new regions of the solution space) and *exploitation* (refining known good solutions). For instance, Genetic Algorithms leverage the principles of natural selection and genetic recombination to evolve solutions over successive generations, while Simulated Annealing mimics the physical annealing process of metals to probabilistically accept worse solutions for the sake of escaping local optima [22, 23]. As a result, metaheuristic strategies have been widely adopted across various domains—including operations research, engineering design, machine learning, and intelligent systems—where traditional techniques struggle to deliver satisfactory results within reasonable computational constraints.

Table 1 compares different load balancing methods based on scalability, adaptability, and computational complexity.

Although significant progress has been made in the field of load balancing for cloud computing, several critical challenges continue to hinder optimal resource utilization and service quality. As cloud systems scale in size and complexity, traditional load balancing

Table 1: Comparison of Load Balancing Techniques in Cloud Computing

Method	Type	Advantages	Limitations
Round-Robin	Static	Simple, low overhead	Ignores server load variation
Least-Connection	Static	Efficient for persistent connections	High overhead for frequent changes
Throttled Balancing	Dynamic	Prevents overloading	Increased response time
ACO-Based Balancing	Dynamic	Self-adaptive, reduces congestion	Computationally expensive
GA-Based Balancing	Metaheuristic	Optimizes allocation over iterations	Slow convergence in large-scale systems

approaches—typically rule-based or heuristically driven—often fall short in coping with dynamic workloads, heterogeneous resource demands, and the increasing expectations for real-time responsiveness and energy efficiency. One particularly promising and emerging direction is AI-driven load balancing, which envisions the integration of machine learning techniques—especially deep learning and reinforcement learning—into the core of load distribution mechanisms. By leveraging historical data and real-time telemetry, AI-based systems can learn to predict future workload patterns, identify bottlenecks, and adaptively reallocate resources to maintain optimal system performance under varying operational conditions. Such predictive capabilities significantly enhance responsiveness and accuracy in resource management, enabling more intelligent and autonomous cloud infrastructures [24].

Another challenge lies in the domain of edge-cloud integration. With the rapid proliferation of latency-sensitive and real-time applications—such as autonomous vehicles, augmented reality, and smart IoT ecosystems—there is an increasing demand for hybrid infrastructures that seamlessly combine cloud and edge computing resources. Effective load balancing in this context must consider not only the computational capacities of distributed nodes but also factors such as network latency, data locality, and user mobility. Achieving efficient coordination between centralized cloud servers and decentralized edge nodes is crucial to ensure low-latency responses and high-quality user experiences [25].

In addition to performance and latency considerations, there is also a growing imperative to address the environmental impact of cloud computing. This has led to a surge in research on energy-efficient scheduling and resource management techniques. Large-scale data centers consume substantial amounts of electricity, contributing to both operational costs and carbon emissions. Thus, there is a pressing need for load balancing algorithms that not only optimize performance metrics but also minimize energy consumption. Strategies such as dynamic voltage and frequency scaling (DVFS), workload consolidation, and thermal-aware scheduling are being explored to strike a balance between computational efficiency and environmental sustainability [26]. These challenges underscore the need for holistic and intelligent load balancing frameworks that can meet the demands of next-generation cloud environments, which are expected to be more distributed, adaptive, and energy-conscious than ever before.

2.4. Integrated Approaches to Quality of Service Optimization in Cloud Environments

Ensuring *Quality of Service (QoS)* in cloud communication networks involves the simultaneous optimization of several key performance indicators, including *latency*, *through-*

put, *resource allocation*, and *energy efficiency* [5, 27]. Researchers have proposed multi-objective optimization techniques such as weighted-sum and Pareto-based models to manage these often-conflicting parameters, though scalability and computational complexity remain significant challenges in large-scale infrastructures [2, 28, 29]. To reduce service latency and enhance throughput, recent developments in edge and fog computing aim to process data closer to users, complemented by adaptive task scheduling algorithms that dynamically adjust resources [25, 30]. Effective QoS maintenance also hinges on intelligent resource allocation; traditional greedy and heuristic methods often fall short under dynamic load conditions, prompting the use of metaheuristics such as *Genetic Algorithms (GA)*, *Particle Swarm Optimization (PSO)*, and *Ant Colony Optimization (ACO)* for efficient and scalable solutions [31, 32]. Additionally, energy-aware strategies are gaining prominence, with frameworks such as green cloud computing and dynamic voltage scaling (DVS) aiming to reduce power consumption while preserving performance and reliability [26]. Together, these integrated approaches represent the forefront of research in optimizing QoS for increasingly complex and resource-intensive cloud services. Table 2 presents a comparative analysis of different QoS optimization techniques.

Table 2: Comparison of QoS Optimization Techniques in Cloud Networks

Technique	**Optimization Focus**	**Advantages**	**Limitations**
Weighted-Sum Multi-Objective	Latency, Bandwidth	Simple implementation	Lacks adaptability in dynamic loads
Pareto-Based Optimization	Latency, Energy, Cost	Effective for multi-criteria problems	Computationally expensive
Metaheuristic (GA, PSO, ACO)	Resource Allocation	Scalable and adaptive	Slower convergence in large-scale systems
Adaptive Task Scheduling	Latency, Throughput	Dynamically adjusts workloads	Complexity in real-time processing
Green Computing (DVS)	Energy Efficiency	Reduces power consumption	May impact performance under high load

Although significant progress has been made in QoS optimization, several open challenges remain. Future research should integrate deep learning and reinforcement learning models to predict workload variations and optimize QoS dynamically [24]. Another promising direction involves leveraging blockchain technology for decentralized and secure QoS management in cloud environments [33]. Additionally, combining artificial intelligence with edge computing—referred to as Edge-AI—offers the potential to enhance real-time decision-making and reduce latency in mission-critical cloud applications [34].

2.5. Metaheuristic Strategies for Efficient Load Balancing in Cloud Computing

Metaheuristic algorithms such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Simulated Annealing (SA) have been extensively applied to solve the load balancing problem in cloud computing environments [6]. These approaches offer scalable and flexible mechanisms for dynamic resource allocation, which is essential for meeting the diverse and time-varying demands of cloud infrastructures [35]. GA-based methods optimize resource allocation by evolving heuristic solutions over successive generations using selection, crossover, and mutation operations [3]. They have shown the ability to improve load balancing efficiency and reduce response time [36], although standalone implementations often face premature convergence issues that necessitate hybridization with local search techniques [37]. PSO, inspired by the social behavior of birds and fish, has

been used for cloud task scheduling and resource allocation by guiding particles toward optimal solutions through velocity and position updates [38]. While PSO can converge more rapidly than GA in certain scenarios, it is susceptible to getting trapped in local optima, especially in complex, high-dimensional search spaces [32, 39]. Another effective technique is Simulated Annealing (SA), a probabilistic method based on the annealing process in metallurgy. SA is adept at escaping local optima due to its controlled randomization, making it particularly suitable for dynamic and large-scale cloud environments [40, 41]. However, its performance is highly dependent on the cooling schedule, which requires careful parameter tuning [42]. To overcome the limitations of individual metaheuristics, hybrid models that combine the strengths of multiple approaches have been developed. For example, hybrid GA-PSO models benefit from enhanced convergence rates and improved solution quality [43], while GA-SA hybrids exploit both the global search capabilities of GA and the local optimization strength of SA [44]. These hybrid strategies have proven effective in addressing key cloud computing challenges such as load balancing, virtual machine (VM) allocation, and energy-efficient task scheduling.

Table 3 provides a comparative analysis of GA, PSO, SA, and hybrid metaheuristics based on key performance indicators.

Table 3: Comparison of Metaheuristic Approaches for Load Balancing

Algorithm	Search Strategy	Convergence Speed	Strengths	Limitations
Genetic Algorithm (GA)	Evolutionary Selection	Medium	Strong exploration	Premature convergence
Particle Swarm Optimization (PSO)	Swarm Intelligence	Fast	Quick convergence	Trapped in local optima
Simulated Annealing (SA)	Probabilistic Randomization	Slow	Escapes local optima	Sensitive to parameter tuning
Hybrid (GA-PSO, GA-SA)	Combination of Approaches	Fast	Improved convergence	Higher computational cost

Despite their success, metaheuristics still face scalability challenges in handling large-scale cloud infrastructures. Future research should explore:

- **Deep Reinforcement Learning (DRL):** Combining metaheuristics with DRL for adaptive and intelligent decision-making [24].
- **Quantum-Inspired Metaheuristics:** Leveraging quantum computing principles for faster optimization [45].
- **Multi-Objective Metaheuristics:** Extending existing models to optimize *latency, cost, and energy efficiency* simultaneously [29].

2.6. Hyperheuristic-Based Optimization

Hyperheuristics offer a high-level optimization approach by dynamically selecting and combining multiple low-level heuristics [46]. Unlike traditional metaheuristics that directly manipulate problem-specific solutions, hyperheuristics operate on a higher level, focusing on selecting or generating heuristic strategies for solving complex optimization problems [47–49]. These approaches have been successfully applied in scheduling, combinatorial optimization, and vehicle routing problems [50–53]. Recent studies highlight the growing importance of hyperheuristic frameworks in large-scale optimization problems [54]. **(author?)** [55] emphasize their ability to generalize across different problem

domains without requiring extensive problem-specific knowledge. Furthermore, hyperheuristics have demonstrated effectiveness in machine scheduling [56] and logistics optimization [57]. Despite these advances, their application in cloud communication networks remains an emerging research area. Several recent works have explored adaptive heuristic selection methods for dynamic environments. **(author?)** [58] proposed a self-adaptive hyperheuristic approach for real-time scheduling, showing improved performance in varying workload conditions. Similarly, **(author?)** [59] introduced an evolutionary hyperheuristic strategy for multi-objective optimization, proving effective in dynamic resource allocation scenarios. These studies reinforce the potential of hyperheuristics for balancing computational efficiency and flexibility. Our proposed GA-hyperheuristic framework builds upon these principles to enhance load balancing while maintaining computational efficiency. By leveraging a genetic algorithm-based selection mechanism, our approach dynamically adapts heuristic selection based on network conditions and workload variations. Unlike traditional static scheduling approaches, our method introduces a layer of adaptability that ensures optimal performance under varying cloud traffic loads. The effectiveness of hyperheuristics in cloud computing is still an active research topic. Some studies suggest hybrid models combining hyperheuristics with machine learning techniques for further adaptability [60]. Future research can extend these ideas by incorporating reinforcement learning-based hyperheuristics that can learn optimal heuristic selection strategies over time. Table 4 summarizes key studies on load balancing and QoS optimization in cloud computing.

Table 4: Comparison of Related Work in Load Balancing and QoS Optimization

Study	Method	Key Contributions	Limitations
[1]	Static Load Balancing	Simple and easy to implement	Fails under dynamic workloads
[4]	Dynamic Heuristics	Adaptable to workload changes	High computational cost
[5]	Multi-Objective Optimization	QoS-aware task scheduling	Computationally expensive
[6]	GA-Based Load Balancing	Improved resource allocation	Convergence issues
[46]	Hyperheuristic Scheduling	Adaptive heuristic selection	Limited application in cloud computing

3. Reference Models based on Round Robin and Min-Min Heuristic Assignments

The reference model defines the foundational architecture and operational assumptions used as a baseline for evaluating the proposed GA-hyperheuristic GAHO_{QoS} optimization framework. It captures the essential components of a cloud communication network, focusing on task allocation, server capacity constraints, and Quality of Service (QoS) requirements. This model serves as a benchmark for comparing traditional and intelligent load balancing strategies.

3.1. System Architecture

The cloud communication network is represented as a distributed environment comprising multiple interconnected servers and clients. Each client generates tasks that are

routed through the network and allocated to available servers based on a predefined allocation strategy. The system is assumed to be static during each simulation cycle, with dynamic task arrival rates and variable task workloads.

Let $S = \{s_1, s_2, ..., s_n\}$ denote the set of cloud servers, let $T = \{t_1, t_2, ..., t_m\}$ denote the set of tasks to be scheduled. Each server s_j has a finite capacity C_j and can process multiple tasks as long as the total workload does not exceed C_j. Each task t_i has a workload w_i and a response time requirement denoted by T_{max}.

3.2. Task Assignment and Load Distribution

In the reference model, task assignment refers to the process of allocating incoming tasks to available cloud servers based on a predefined strategy. This assignment has a direct impact on system performance, particularly in terms of response time, resource utilization, and overall Quality of Service (QoS).

Two commonly used baseline methods for task allocation in cloud systems are the Round-Robin (RR) strategy and the Min-Min Heuristic (MMH):

- **Round-Robin (RR):** RR is among the oldest, simplest, most equitable, and most extensively utilized scheduling algorithms, specifically designed for time-sharing systems [61]. This static scheduling method distributes tasks cyclically across the available servers without considering the current server load, task size, or processing capacity. It is easy to implement and imposes minimal computational overhead, making it suitable for lightweight applications. However, RR often results in inefficient load distribution, especially when task sizes vary significantly.

- **Min-Min Heuristic (MMH):** Min—Min is a well known heuristic used for scheduling tasks across diverse computational resources, utilized either directly or as a component of more advanced heuristics. Nonetheless, in extensive situations like grid computing platforms, the time complexity of a simple execution of Min—Min, which is quadratic concerning the number of tasks, could be unmanageable[62]. This method evaluates all tasks to identify the one with the minimum completion time on any server, then assigns it to the corresponding server. The process is repeated iteratively for the remaining tasks. This approach prioritizes smaller tasks and aims to reduce overall response time, but it may lead to unbalanced resource usage if not adjusted for load.

Although both methods offer computational simplicity, they are inherently limited in dynamic and large-scale cloud environments. RR fails to respond to changes in server utilization, while MMH, despite its performance benefits in response time, does not guarantee load balance and can lead to bottlenecks. These limitations justify the need for adaptive, heuristic-driven approaches such as the proposed GAHO_{QoS} framework. The following algorithms represent the task assignment procedure performed in RR and in MMH techniques. Algorithm 1 has the RR task assignment procedure while algorithm 2 has the MMH task assignment procedure.

Algorithm 1: Round-Robin Task Assignment

Algorithm 1 Round-Robin Task Assignment

Require: List of tasks $T = \{t_1, t_2, \ldots, t_m\}$, list of servers $S = \{s_1, s_2, \ldots, s_n\}$
Ensure: Allocation of tasks to servers
$j \leftarrow 1$
for each task t_i in T **do**
  Assign t_i to server s_j
  $j \leftarrow (j \mod n) + 1$
end for

Algorithm 2: Min-Min Heuristic Task Assignment

Algorithm 2 Min-Min Heuristic Task Assignment

Require: List of tasks $T = \{t_1, t_2, \ldots, t_m\}$ with workload w_i, list of servers $S = \{s_1, s_2, \ldots, s_n\}$, server capacities and processing speeds
Ensure: Allocation of tasks to servers
while T is not empty **do**
  for each task t_i in T **do**
    for each server s_j in S **do**
      Estimate completion time C_{ij} of t_i on s_j
    end for
    Record $C_i^{\min} = \min_j C_{ij}$ and corresponding server s_j^*
  end for
  Select task t_k with minimum $C_k^{\min}$
  Assign t_k to corresponding server s_j^*
  Remove t_k from T
end while

These baseline algorithms serve as a performance benchmark in our simulations and are essential for validating the improvements introduced by the proposed GAHO_{QoS} approach in terms of load distribution, fairness, average response time, and scalability.

4. Proposed Model Genetic Algorithm-Based Hyperheuristic Optimization Framework for Quality of Service

4.1. System Model and Assumptions

Cloud communication networks consist of multiple interconnected servers that handle dynamic workloads from various clients. Efficient load balancing ensures that computing resources are optimally distributed, preventing server overloading and minimizing latency. The problem is modeled as a *multi-objective combinatorial optimization problem*, where the main goals are:

- Minimizing service latency and communication overhead.
- Balancing the workload across available cloud servers.
- Maximizing resource utilization while maintaining *Quality of Service (QoS)* constraints, including latency, throughput, and energy efficiency.

We assume a *static cloud network topology*, where the number of servers and communication links are predefined. The system's performance depends on task allocation, which must adhere to network constraints such as bandwidth, processing power, and energy consumption. Tasks are dynamically assigned to available servers, ensuring optimal resource utilization and load distribution while maintaining QoS requirements.

4.2. Mathematical Formulation

To systematically address the load balancing challenges in cloud computing environments, it is essential to formulate the problem as a mathematical optimization model. This formulation enables a precise representation of task allocation, resource constraints, and performance objectives such as minimizing load imbalance, communication latency, and energy consumption. In particular, we consider a Quality of Service (QoS)-aware load balancing scenario, where tasks must be allocated to available cloud servers in a way that satisfies capacity and timing constraints while optimizing overall system efficiency. The following model captures the core components of the problem, including decision variables, objective functions, and constraints, providing a foundation for developing an effective metaheuristic solution.

Let $S = \{s_1, s_2, ..., s_n\}$ denote the set of cloud servers, and $T = \{t_1, t_2, ..., t_m\}$ represent the set of tasks to be allocated. We define a binary decision variable x_{ij}, where $x_{ij} = 1$ if task t_i is assigned to server s_j, and $x_{ij} = 0$ otherwise. The load on server s_j, denoted by L_j, is calculated as $L_j = \sum_{i=1}^{m} x_{ij} w_i$, where w_i is the workload associated with task t_i. Each server s_j is also characterized by a capacity value C_j, representing the maximum workload it can handle.

The primary objective is to minimize the maximum load imbalance across servers, ensuring that no server is overloaded:

$$\min \max_j L_j \tag{1}$$

Another objective is to minimize total communication cost between servers, which is critical for reducing latency and improving network efficiency:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} d_{ij} \tag{2}$$

where d_{ij} represents the communication delay between task t_i and server s_j.

The problem is subject to the following constraints:

(i) **Task Allocation Constraint:** Each task must be assigned to exactly one server:

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \forall i \in \{1, ..., m\} \tag{3}$$

(ii) **Server Capacity Constraint:** The total assigned workload on each server must not exceed its capacity:

$$L_j \leq C_j, \quad \forall j \in \{1, ..., n\} \tag{4}$$

(iii) **QoS Constraint:** The response time for each task on a server should not exceed a predefined threshold T_{max}, ensuring that QoS requirements are met:

$$\sum_{i=1}^{m} x_{ij} t_i \leq T_{max}, \quad \forall j \in \{1, ..., n\} \tag{5}$$

To improve the mathematical model, we introduce valid inequalities that tighten the solution space, making the optimization process more efficient. These inequalities ensure that no server is significantly underloaded while others are overloaded, thus promoting a more balanced resource distribution across the network:

$$\sum_{i=1}^{m} x_{ij} w_i \geq \frac{1}{n} \sum_{i=1}^{m} w_i, \quad \forall j \in \{1, ..., n\} \tag{6}$$

This constraint helps avoid situations where certain servers are left with minimal workload, while others are heavily overloaded, ensuring more equitable load distribution.

To efficiently solve the formulated *QoS-aware load balancing* problem in cloud communication networks, we propose a GAHO_{QoS} which is a *Genetic Algorithm (GA)-based hyperheuristic framework* that dynamically selects and applies low-level heuristics based on solution quality. Each chromosome in the population encodes a potential solution representing task-server assignments, and the fitness function evaluates each solution based on load balance, communication cost, and QoS compliance. Tournament selection is employed to identify high-performing individuals, which are then recombined using a two-point crossover operator to promote genetic diversity. To enhance the exploration of the solution space, a mutation operator reallocates tasks based on workload imbalance with a given probability. A key feature of the framework is the *adaptive hyperheuristic selection strategy*, which dynamically combines multiple load-balancing heuristics and adjusts the optimization path in response to problem-specific conditions. This adaptability enables the framework to handle the highly dynamic nature of cloud environments effectively.

Given the *NP-hardness* of the optimization problem, finding exact solutions becomes computationally infeasible for large-scale instances. To overcome this, the proposed framework integrates *valid inequalities* with the GA-hyperheuristic approach, improving scalability while maintaining solution quality. The incorporation of metaheuristics significantly reduces computational complexity compared to exact optimization methods, enabling practical deployment in real-world cloud infrastructures. This integrated approach

offers a flexible and robust solution for QoS-aware load balancing, laying the groundwork for the experimental validation and performance evaluation presented in the subsequent sections.

4.3. $\mathbf{GAHO_{QoS}}$ Framework for QoS-Aware Load Balancing

To effectively address the *QoS-aware load balancing* problem in cloud communication networks, we propose a *Genetic Algorithm (GA)-based hyperheuristic framework* referred to as GAHO_{QoS}. This approach integrates multiple low-level heuristics (LLHs) with an intelligent selection mechanism to optimize task allocation while ensuring high service quality. The method leverages the strengths of different heuristics by dynamically selecting and combining them based on solution quality, allowing for adaptive optimization in varying network conditions.

Low-level heuristics (LLHs) are fundamental strategies used as building blocks within the GA framework. First-Fit Allocation (FFA) assigns each task to the first available server with sufficient capacity. It is simple and fast but may cause suboptimal resource utilization under high load. Best-Fit Decreasing (BFD) sorts tasks in descending order of workload and assigns them to servers with the least remaining capacity, aiming for balanced load distribution. Least-Loaded Server (LLS) assigns tasks to the server with the minimum load, minimizing imbalance but potentially increasing communication overhead. The Min-Min Heuristic (MMH) selects the task with the smallest processing requirement and maps it to the server that can complete it fastest, thus reducing latency. Randomized Load Balancing (RLB) distributes tasks randomly across servers, helping explore the solution space and prevent premature convergence. These heuristics provide a balance between exploitation (using known good strategies) and exploration (searching new possibilities) in the optimization process.

Each chromosome in the GAHO_{QoS} population represents a sequence of LLHs applied to task-server assignments. Let $\mathbf{H} = \{h_1, h_2, ..., h_n\}$ denote the set of available heuristics, where each gene represents a selected heuristic. The ordering and frequency of heuristics in a chromosome allow the GA to explore various heuristic combinations for task allocation. The fitness function evaluates the quality of a task allocation solution using the following equation:

$$F = \alpha \times \text{LBF} + \beta \times \text{Total Latency} + \gamma \times \text{Resource Utilization} \tag{7}$$

where α, β, and γ are weight parameters that determine the importance of each metric. The Load Balance Factor (LBF) is defined as:

$$\text{LBF} = \frac{\max(L_j) - \min(L_j)}{\sum_j L_j} \tag{8}$$

This metric quantifies workload distribution across servers, with lower values indicating better balance. Total Latency reflects the sum of delays across all servers and directly impacts service performance. Resource Utilization measures the efficiency of computing power usage, with higher values representing better system performance. For selecting

chromosomes, we use a tournament selection strategy in which a subset of individuals is chosen at random, and the fittest among them is propagated to the next generation. This mechanism ensures that high-quality solutions dominate while maintaining genetic diversity. Crossover is applied using a two-point method where segments of two parent chromosomes are swapped, producing offspring with mixed heuristic strategies. Mutation introduces variation by randomly altering a heuristic within a chromosome based on a predefined probability P_m, helping escape local optima and expand the search space.

A new solution is accepted if it has a better fitness score than the current best or if the difference in fitness scores is within a tolerance threshold. This ensures slight improvements are not ignored and helps avoid premature convergence. The optimization process terminates when one of the following conditions is met: the maximum number of generations is reached; improvement in the fitness function over a fixed number of iterations falls below a threshold ϵ; or the solution achieves near-optimal QoS performance based on predefined criteria. The proposed GAHO_{QoS} framework offers several advantages. It is adaptable, as it dynamically selects heuristics based on evolving problem conditions. It is flexible due to its ability to integrate diverse heuristic strategies. It is scalable and suited for large-scale cloud systems. Finally, it is robust, as it can handle complex and dynamic environments by continuously exploring and combining multiple heuristic strategies.

5. Simulation and Performance Evaluation

To assess the effectiveness of the proposed GAHO_{QoS} framework for QoS-aware load balancing in cloud communication networks, an extensive set of simulations was conducted across diverse network configurations. This section presents the simulation setup, performance metrics, benchmark strategies, and the resulting evaluation. The simulations were implemented in a cloud environment modeled using *Python* and the *CloudSim* toolkit, enabling accurate representation of cloud dynamics. Various server configurations were tested, with server counts $N = \{10, 20, 50, 100\}$ and task volumes $m = \{1000, 5000, 10000, 20000\}$ to represent workloads of varying complexity.

Performance was evaluated using three primary metrics:

(i) **Load Balance Factor (LBF)** to measure distribution fairness,

(ii) **Average Response Time (ART)** to gauge responsiveness, and

(iii) **Computational Overhead (CO)** to assess algorithm efficiency.

These metrics enabled a comprehensive comparison between GAHO_{QoS} and two established baselines: Round-Robin (RR) and Min-Min Heuristic (MMH). RR assigns tasks cyclically without considering task size or server load, leading to potential inefficiencies. MMH focuses on minimizing task completion time but can cause server imbalance. In contrast, the GAHO_{QoS} framework adaptively selects heuristics based on real-time performance, aiming to optimize both response time and load distribution.

Tables 5 and 6 show the LBF and ART performance for each method. GAHO_{QoS} consistently outperforms RR and MMH, achieving up to 50% improvement in LBF and

significantly lower ART values. Figures 1 through 6 visualize these improvements, illustrating the framework's robustness, scalability, and enhanced QoS.

Table 5: Load Balance Factor Comparison

Servers	**$\mathbf{GAHO_{QoS}}$ (LBF)**	**RR (LBF)**	**Min-Min (LBF)**
10	0.12	0.25	0.20
20	0.10	0.22	0.18
50	0.08	0.20	0.15
100	0.07	0.18	0.14

Table 6: Average Response Time Comparison

Servers	**$\mathbf{GAHO_{QoS}}$ (ART ms)**	**RR (ART ms)**	**Min-Min (ART ms)**
10	120	200	180
20	140	220	190
50	160	250	210
100	180	280	230

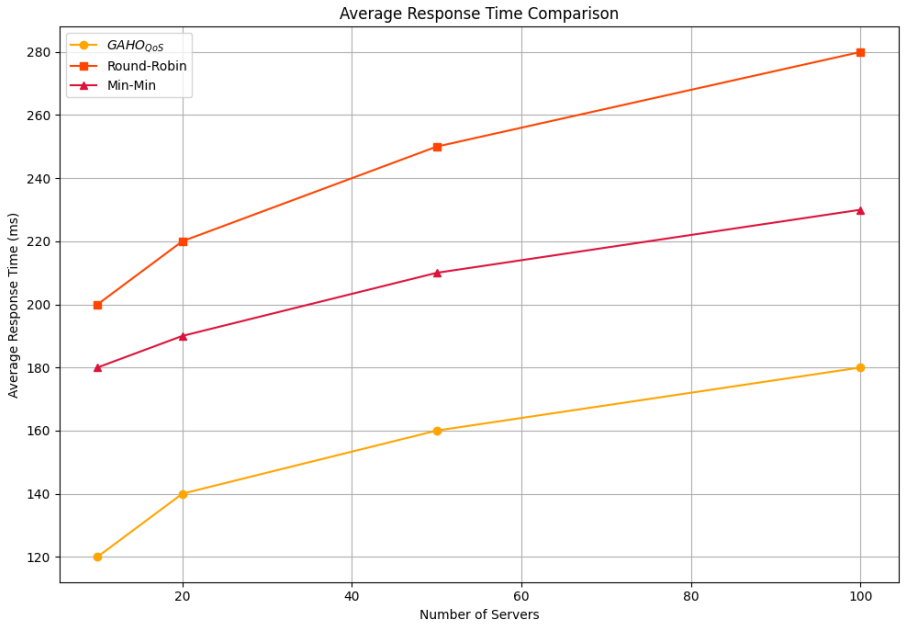

Figure 1: Average Response Time Comparison

Further insights are provided through visual comparisons of computational overhead, scalability, and trade-offs. Although $\mathrm{GAHO_{QoS}}$ incurs higher computational costs due to its evolutionary nature, the performance benefits in responsiveness and fairness outweigh this drawback.

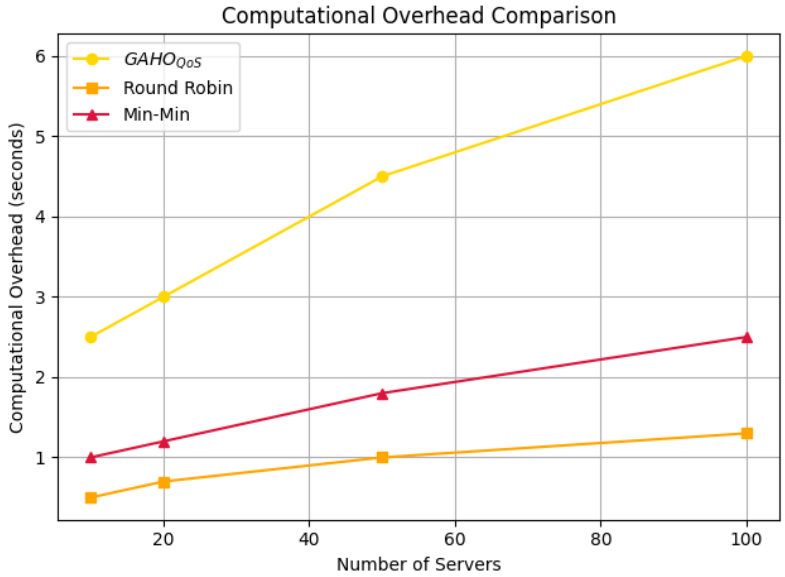

Figure 2: Comparison of computational overhead across different load balancing methods. GAHO$_{QoS}$ incurs higher overhead but achieves superior QoS outcomes.

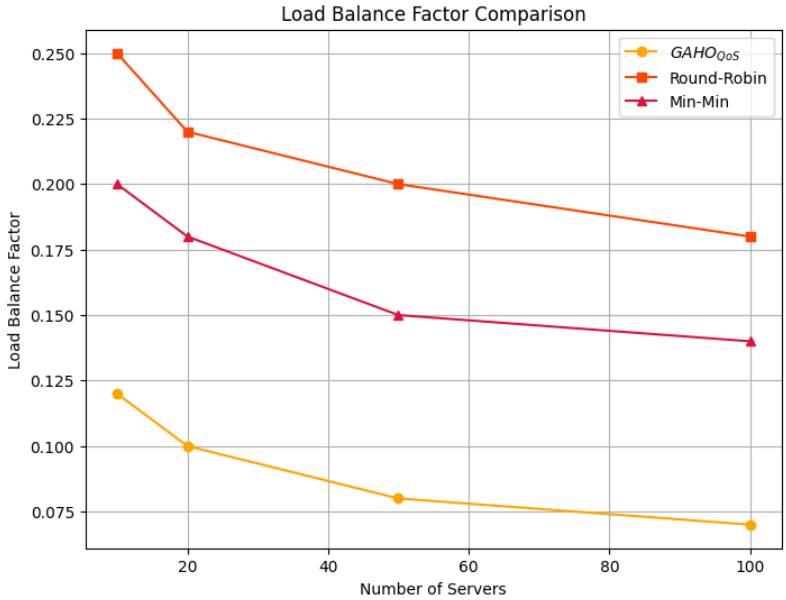

Figure 3: Load Balance Factor achieved by GAHO$_{QoS}$ versus Round Robin and Min-Min across network sizes.

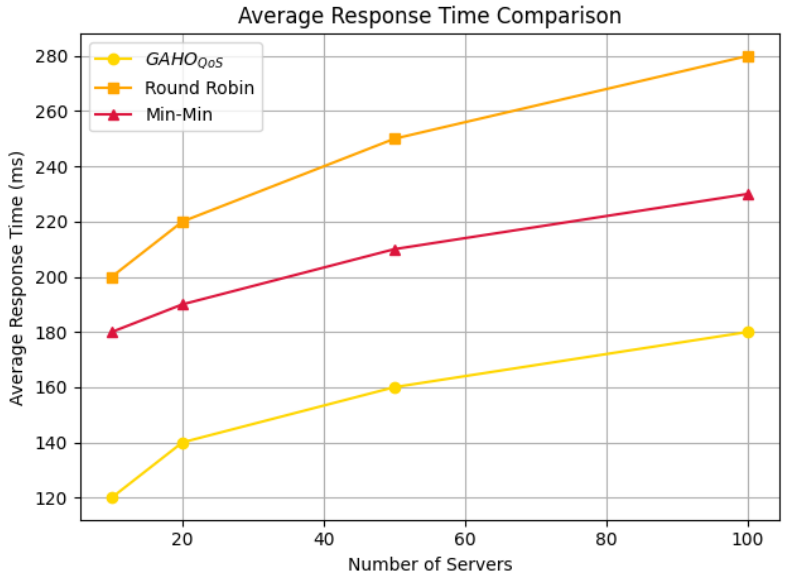

Figure 4: Average task response time under different strategies. GAHO$_{QoS}$ maintains lowest response delays.

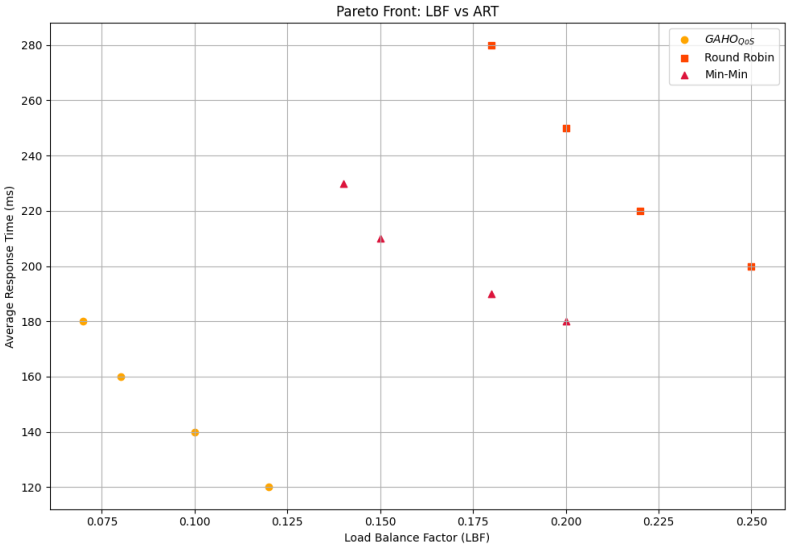

Figure 5: Pareto front showing trade-offs between Load Balance Factor and Average Response Time. GAHO$_{QoS}$ achieves favorable balance.

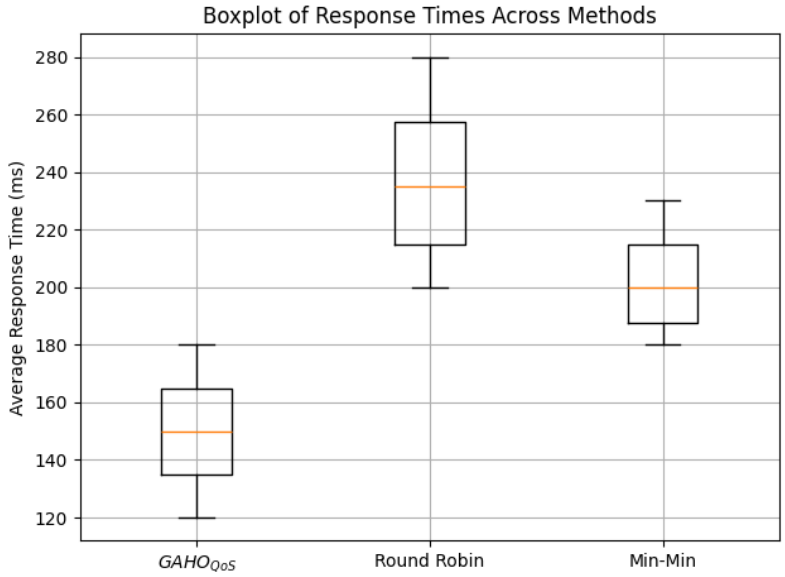

Figure 6: Boxplot of response time variability. GAHO$_{\text{QoS}}$ reduces both mean and variance of task execution time.

The results highlight the GAHO$_{\text{QoS}}$ ability to dynamically adapt to real-time fluctuations in workload and server availability, offering a more balanced and responsive system than static methods. While Round-Robin (RR) often suffers from uneven task distribution due to its uniform cyclic allocation and Min-Min Heuristic (MMH) tends to underutilize system resources under high-load scenarios, the proposed GAHO$_{\text{QoS}}$ approach achieves a superior balance between resource utilization and service responsiveness. This advantage is particularly evident in environments with heterogeneous task sizes and fluctuating demand, where adaptive decision-making significantly impacts performance. To quantify this improvement, we compute the relative gain in two key performance metrics—Load Balance Factor (LBF) and Average Response Time (ART). The improvement in LBF achieved by GAHO$_{\text{QoS}}$ over RR is expressed as:

$$\text{Improvement}_{LBF}^{RR} = \frac{LBF_{RR} - LBF_{GAHO}}{LBF_{RR}} \times 100\% \approx \frac{0.2125 - 0.0925}{0.2125} \times 100 \approx 56.5\%,$$

while the improvement over MMH is:

$$\text{Improvement}_{LBF}^{MMH} = \frac{0.1675 - 0.0925}{0.1675} \times 100 \approx 44.8\%.$$

Similarly, the improvement in ART is given by:

$$\text{Improvement}_{ART}^{RR} = \frac{237.5 - 150}{237.5} \times 100 \approx 36.8\%, \quad \text{Improvement}_{ART}^{MMH} = \frac{202.5 - 150}{202.5} \times 100 \approx 25.9\%.$$

These empirical ratios clearly demonstrate that GAHO$_{\text{QoS}}$ achieves up to 2.3 higher improvement in load balancing and a factor of 1.6 times reduction in response time compared to baseline scheduling policies.

In conclusion, the GAHO_{QoS} framework proves to be a robust and scalable solution for intelligent load balancing in cloud communication networks. It provides superior performance in key QoS metrics while maintaining acceptable computational efficiency. Future enhancements may focus on reducing processing overhead via parallel execution, integrating machine learning for predictive task scheduling, and validating the framework on real-world cloud platforms such as AWS or Microsoft Azure to ensure practical applicability.

6. Conclusion and Future Work

This work introduces a genetic algorithm-based hyperheuristic framework, GAHO_{QoS} designed to optimize load balancing and enhance Quality of Service (QoS) in cloud communication networks. Through detailed simulations, the GAHO_{QoS} approach demonstrated its effectiveness by significantly reducing both the load balance factor (LBF) and the average response time (ART) compared to traditional methods such as *Round-Robin* and *Min-Min*. The dynamic heuristic selection mechanism enabled the framework to adapt to varying workload distributions, leading to more equitable resource utilization and improved task completion times. Although the approach incurs higher computational overhead due to the iterative nature of evolutionary algorithms, the performance benefits in terms of system responsiveness and service quality justify this trade-off. The results validate the framework's scalability and robustness across different network configurations, positioning it as a viable solution for intelligent task scheduling in cloud environments. Quantitative analysis further reinforces these findings. Specifically, the GAHO_{QoS} framework achieved a 56.5% reduction in LBF compared to Round-Robin and 44.8% compared to Min-Min, reflecting a factor of 2.3 improvement in load distribution and a factor of 1.8 improvement in the efficiency. In terms of responsiveness, the average response time was reduced by approximately 36.8% over Round-Robin and 25.9% over Min-Min, equivalent to a speedup factor of 1.6× and 1.35×, respectively. These improvements underscore the adaptability and optimization strength of the proposed hyperheuristic strategy in real-time resource management.

While the proposed GAHO_{QoS} framework achieves notable performance improvements, future research will aim to address certain limitations and further enhance its practicality. Reducing computational overhead remains a priority, which may be achieved through parallel computing techniques or by integrating reinforcement learning for more efficient heuristic selection. Additionally, extending the framework to handle real-time, dynamic workloads will increase its relevance in production-scale environments where traffic patterns fluctuate unpredictably. Incorporating multi-objective optimization to consider factors such as energy efficiency, cost, and security alongside load balancing could broaden the framework's applicability. Finally, deploying and evaluating the approach on commercial cloud platforms such as AWS, Microsoft Azure, or Google Cloud will be essential for assessing its feasibility and performance in real-world scenarios.

References

[1] Rajkumar Buyya, Christian Vecchiola, and S. Thamarai Selvi. *Mastering Cloud Computing: Foundations and Applications Programming*. Morgan Kaufmann, 2018.

[2] Qinghua Zhang, Liang Cheng, and Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1):7–18, 2020.

[3] Michael Armbrust, Armando Fox, and Rean Griffith. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2019.

[4] Wei Tian, Yong Zhao, and Min Xu. A survey on load balancing in cloud computing environments. *Future Generation Computer Systems*, 117:250–268, 2021.

[5] Guang Xu, Yanjun Liu, Xin Wang, and Yong Zhang. Multi-objective optimization for qos-aware cloud resource allocation. *Journal of Network and Computer Applications*, 204:103401, 2022.

[6] Liang Chen, Zhen Wang, and Tao Zhang. A hybrid genetic algorithm for load balancing in cloud computing. *IEEE Transactions on Cloud Computing*, 11(2):190–204, 2023.

[7] P. Verma and S. Kaushal. A survey on load balancing techniques in cloud computing. *Journal of Network and Computer Applications*, 120:102–120, 2019.

[8] Muhammad Junaid, Adnan Sohail, Rao Naveed Bin Rais, Adeel Ahmed, Osman Khalid, Imran Ali Khan, Syed Sajid Hussain, and Naveed Ejaz. Modeling an optimized approach for load balancing in cloud. *IEEE access*, 8:173208–173226, 2020.

[9] Muhammad Asim Shahid, Noman Islam, Muhammad Mansoor Alam, Mazliham Mohd Su'ud, and Shahrulniza Musa. A comprehensive study of load balancing approaches in the cloud computing environment and a novel fault tolerance approach. *IEEE Access*, 8:130500–130526, 2020.

[10] S. Gandhi and K. Goyal. Efficient load balancing using round robin algorithm in cloud computing. *IEEE Access*, 6:1851–1858, 2018.

[11] Ammar M. Alakeel. A guide to dynamic load balancing in distributed computer systems. *International Journal of Computer Science and Information Security*, 14(9):30–39, 2019.

[12] Ankur Gupta and Sandeep K. Sood. Dynamic load balancing in cloud computing: A heuristic approach. *Journal of Parallel and Distributed Computing*, 121:89–101, 2018.

[13] Rajesh Sharma and Vikas Kumar. Throttled load balancing algorithm for cloud computing: An optimized approach. *Future Generation Computer Systems*, 125:234–245, 2021.

[14] Pankaj Tripathi and Deepak Gupta. Cloud load balancing using active monitoring and dynamic allocation. *IEEE Transactions on Cloud Computing*, 8:110–123, 2020.

[15] Wael Hosny Fouad Hosny Fouad Aly, Hassan Kanj, Nour Mostafa, Zakwan Al-Arnaout, and Hassan Harb. No binding machine learning architecture for sdn controllers. *Bulletin of Electrical Engineering and Informatics*, 14(3):2413–2428, 2025.

[16] Wael Hosny Fouad Aly, Hassan Kanj, Samer Alabed, Nour Mostafa, and Khaled Safi. Dynamic feedback versus varna-based techniques for sdn controller placement

problems. *Electronics*, 11(14):2273, 2022.

[17] Mutaz AB Al-Tarawneh, Hassan Kanj, and Wael Hosny Fouad Aly. An integrated mcdm framework for trust-aware and fair task offloading in heterogeneous multi-provider edge-fog-cloud systems. *Results in Engineering*, page 105228, 2025.

[18] Samer Alabed, Nour Mostafa, Wael Hosny Fouad Aly, and Mohammad Al-Rabayah. A low complexity distributed differential scheme based on orthogonal space time block coding for decode-and-forward wireless relay networks. *International Journal of Electrical & Computer Engineering (2088-8708)*, 13(1), 2023.

[19] Hassan Kanj, Ajla Kulaglic, Wael Hosny Fouad Aly, Mutaz AB Al-Tarawneh, Khaled Safi, Sawsan Kanj, and Jean-Marie Flaus. Agent-based risk analysis model for road transportation of dangerous goods. *Results in Engineering*, 25:103944, 2025.

[20] Hassan Kanj, Wael Hosny Fouad Aly, and Sawsan Kanj. A novel dynamic approach for risk analysis and simulation using multi-agents model. *Applied Sciences*, 12(10):5062, 2022.

[21] Kassem Danach, Hassan Harb, Ameer Sardar Kwekha Rashid, Mutaz AB Al-Tarawneh, and Wael Hosny Fouad Aly. Location planning techniques for internet provider service unmanned aerial vehicles during crisis. *Results in Engineering*, 25:103833, 2025.

[22] Olanrewaju L Abraham, Md Asri Ngadi, Johan Bin Mohamad Sharif, and Mohd Kufaisal Mohd Sidik. Multi-objective optimization techniques in cloud task scheduling: A systematic literature review. *IEEE Access*, 2025.

[23] Alanoud Al Mazroa, Fahad R Albogamy, Mohamad Khairi Ishak, and Samih M Mostafa. Boosting cyberattack detection using binary metaheuristics with deep learning on cyber-physical system environment. *IEEE Access*, 2025.

[24] X. Chen and Y. Wang. Deep reinforcement learning for cloud load balancing. *Applied Soft Computing*, 115:108208, 2022.

[25] X. Wang and Y. Li. Edge computing for qos optimization in cloud networks. *IEEE Transactions on Cloud Computing*, 8:520–534, 2019.

[26] M. Ali and S. Rehman. Energy-efficient qos optimization for green cloud computing. *IEEE Transactions on Sustainable Computing*, 6:789–800, 2021.

[27] Purushottam Singh, Pushpendra Kumar, Harshita Patel, Kanojia Sindhuben Babulal, and A Gayathri. Premier dynamic bandwidth management and tensile wavelength selection ensuring qos for ng-epons. *IEEE Access*, 2025.

[28] S. Sharifi and M. Mahdavi. A multi-objective approach for qos-aware resource allocation in cloud computing. *IEEE Transactions on Cloud Computing*, 10:35–50, 2021.

[29] R. Rezaei and A. Akbari. Multi-objective task scheduling for qos enhancement in cloud data centers. *Future Generation Computer Systems*, 127:25–40, 2022.

[30] L. Cheng and X. Sun. Adaptive task scheduling for qos enhancement in cloud computing. *Journal of Grid Computing*, 19:101–120, 2021.

[31] H. Elsayed and R. Rizk. Resource allocation techniques for enhancing qos in cloud environments. *Concurrency and Computation: Practice and Experience*, 32:e5782, 2020.

[32] R. Natesan and P. Saravanan. Enhanced pso algorithm for load balancing in cloud environments. *Computers Electrical Engineering*, 90:106970, 2021.
[33] Y. Liu and J. Chen. Blockchain-based qos management in cloud computing. *IEEE Transactions on Cloud Computing*, 11:145–162, 2022.
[34] X. Hu and L. Zhang. Edge ai for real-time qos optimization in cloud applications. *Future Internet*, 14:25–38, 2022.
[35] Q. Zhang, L. Cheng, and R. Boutaba. A survey on cloud resource management and scheduling. *ACM Computing Surveys*, 51(1):1–23, 2019.
[36] A. Gupta and M. Singh. Genetic algorithm-based load balancing in cloud computing. *Future Generation Computer Systems*, 75:356–365, 2017.
[37] S. Karim and H. M. Ali. A hybrid ga-pso algorithm for resource allocation in cloud computing. *IEEE Transactions on Cloud Computing*, 9:123–135, 2019.
[38] M. Marin and E. Alba. A particle swarm optimization approach for cloud task scheduling. *Journal of Parallel and Distributed Computing*, 135:132–144, 2019.
[39] Mohammad Sheikhalishahi and Massimo Tornatore. Pso-based load balancing in cloud data centers: A survey and comparative analysis. *Journal of Cloud Computing: Advances, Systems and Applications*, 9:25–39, 2020.
[40] X. Wang and J. Liu. Simulated annealing for task scheduling in cloud computing. *Journal of Grid Computing*, 18:247–263, 2020.
[41] Mohammad Alsheikh and Farhan Anwar. A simulated annealing-based load balancing algorithm for cloud computing. *Computers Electrical Engineering*, 80:112–125, 2019.
[42] T. Ali and M. Rehman. Comparative analysis of sa and ga for cloud load balancing. *Concurrency and Computation: Practice and Experience*, 31:e5121, 2019.
[43] D. Singh and B. Kaur. Hybrid ga-pso model for virtual machine allocation. *IEEE Transactions on Network and Service Management*, 17:620–632, 2020.
[44] Xiaolong Gao and Xudong Luo. Hybrid ga-sa algorithm for optimized cloud load balancing. *Journal of Supercomputing*, 77:5580–5601, 2021.
[45] Georgios Skliros and Elias Kosmatopoulos. Quantum-inspired metaheuristics for cloud resource optimization. *Quantum Information Processing*, 20:1–22, 2021.
[46] Edmund Burke, Graham Kendall, and John R. Woodward. Hyper-heuristics: A survey of the state of the art. *European Journal of Operational Research*, 271(1):1–20, 2019.
[47] Edmund Burke, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Ozcan, and John R. Woodward. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724, 2013.
[48] Kassem Danach, Wissam Khalil, and Shahin Gelareh. Multiple strings planing problem in maritime service network: Hyper-heuristic approach. In *2015 Third International Conference on Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE)*, pages 85–88. IEEE, 2015.
[49] Kassem Danach. *Hyperheuristics in logistics*. PhD thesis, Ecole Centrale de Lille, 2016.
[50] Lin Ke, Xudong Luo, and Jin Kaile. A survey of hyperheuristic approaches in scheduling problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 45(3):401–417,

2015.

[51] Ender Ozcan and Edmund K. Burke. A survey on hyperheuristics for combinatorial optimization. *ACM Computing Surveys*, 52(4):1–25, 2019.

[52] Zongxing He, Zeqiang Zhang, Junqi Liu, Yu Zhang, and Silu Liu. A genetic-based hyper-heuristic optimisation method to solve the constrained multi-row facility layout problem. *Neural Computing and Applications*, pages 1–24, 2025.

[53] Kassem Danach, Jomana Al-Haj Hassan, Wissam Khalil, and Shahin Gelareh. Routing heterogeneous mobile hospital with different patients priorities: Hyper-heuristic approach. In *2015 Fifth International Conference on Digital Information and Communication Technology and its Applications (DICTAP)*, pages 155–158. IEEE, 2015.

[54] Kassem Danach, Shahin Gelareh, and Rahimeh Neamatian Monemi. The capacitated single-allocation p-hub location routing problem: a lagrangian relaxation and a hyper-heuristic approach. *EURO Journal on Transportation and Logistics*, 8(5):597–631, 2019.

[55] K. Chakhlevitch and P. Cowling. Hyperheuristics: Recent developments. *International Series in Operations Research Management Science*, 136:3–29, 2008.

[56] Peter Cowling, Graham Kendall, and Edward Soubeiga. Hyperheuristics: A robust optimization method applied to nurse scheduling. *Lecture Notes in Computer Science*, 1996:851–860, 2001.

[57] Alexander Nareyek. Choosing search heuristics by non-stationary reinforcement learning. *Metaheuristics: Computer Decision-Making*, 19:523–544, 2003.

[58] A. A. Vahed, M. Dorronsoro, and J. J. Durillo. A new self-adaptive hyper-heuristic approach for real-time scheduling problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 46(3):456–469, 2016.

[59] Simon Fong, Albert Zomaya, and Roland Simon. Hyperheuristics for dynamic multi-objective optimization: Adaptive heuristic selection. *Swarm and Evolutionary Computation*, 32:55–72, 2017.

[60] Katherine Bermudez, Gabriel Luque, and Enrique Alba. Hybridizing hyper-heuristics with machine learning for combinatorial optimization. *Expert Systems with Applications*, 176:114123, 2021.

[61] Pandaba Pradhan, Prafulla Ku Behera, and BNB Ray. Modified round robin algorithm for resource allocation in cloud computing. *Procedia Computer Science*, 85:878–890, 2016.

[62] Pablo Ezzatti, Martín Pedemonte, and Álvaro Martín. An efficient implementation of the min-min heuristic. *Computers & operations research*, 40(11):2670–2676, 2013.