Graph Neural Networks-Based Multi-Objective Optimization for Energy-Efficient Big Data Processing in Cloud Computing

Ke Hu

***Abstract*-The explosive expansion of big-data workloads in cloud platforms has intensified challenges surrounding both resource utilization and power draw. This work proposes a novel multi-objective optimization framework that employs graph neural networks (GNNs) to capture the intricate task–resource relationships and leverages a tailored policy-gradient (PG) algorithm to learn allocation strategies in continuous action spaces. A composite reward function—accounting for energy usage, task completion latency, and additional service metrics—guides the learning process, enabling an explicit trade-off between energy efficiency and overall performance. On synthetic workloads our approach cuts energy consumption by 35 %, 28 %, and 19 % relative to GH, MOGA, and DRL baselines, respectively. Comparable gains (31%–39%) are achieved on the Google Cluster Trace dataset. Moreover, the framework maintains competitive makespan and average completion time while distributing resources fairly. These findings demonstrate the effectiveness of the proposed method in simultaneously delivering energy savings and high performance for cloud-based big-data processing.**

***Index Terms*-Cloud computing, big data processing, energy efficiency, Policy Gradient (PG) Algorithm, multi-objective optimization**

I. Introduction

The pervasive growth of big data has fundamentally reshaped how organizations process, analyze, and extract value from vast quantities of structured and unstructured information. However, the inherent characteristics of big data-notably its sheer volume, rapid velocity, and diverse variety-present formidable challenges to conventional computing infrastructures [1]. Traditional on-premises systems frequently prove inadequate, lacking the requisite storage capacity, bandwidth, and computational power to manage big data effectively at scale. Furthermore, sophisticated data analytics and advanced visualization techniques, essential for deriving critical insights, demand specialized resources that often surpass the capabilities of local computing environments [2]. In response to these limitations, cloud computing has emerged as a transformative paradigm, offering dynamically scalable storage, high-capacity bandwidth, powerful analytical tools, and sophisticated data visualization platforms [3]. By harnessing the elastic, on-demand resources inherent to cloud computing, organizations can more adeptly address the complexities of big data processing, thereby unlocking the full potential of their data assets [4].

Manuscript received March 11, 2025; revised July 22, 2025.

Ke Hu is a Lecturer in Laboratory Construction Management and Operation Center, Nanyang Institute of Technology, Nanyang 473004, China (corresponding author to provide e-mail: 2091004@nyist.edu.cn).

Virtualization in cloud data centers aggregates physical hardware into an abstracted pool, enabling fine-grained allocation of compute, storage, and network resources to dynamic big-data workloads [5]. However, widespread cloud adoption introduces new hurdles—notably load balancing, resource contention, and, critically, energy consumption, issues further complicated by mobile-cloud scenarios [6]. Static heuristics and rule-based optimizers struggle to keep pace with the heterogeneity and volatility of modern cloud environments [7]. The problem is exacerbated by ever-expanding datasets with diverse processing needs and stringent QoS requirements [8], underscoring the need for adaptive, intelligent resource-management strategies.

Swarm-intelligence algorithms have shown promise for such optimization tasks. Inspired by collective behaviors seen in nature, techniques such as particle swarm optimization (PSO) and ant colony optimization (ACO) can explore large search spaces to produce near-optimal schedules [9], [10]. Their relative simplicity and robustness make them attractive for multi-objective scenarios [11], and hybrid variants further enhance trade-off handling among performance, energy, and QoS metrics [12]. Nevertheless, conventional swarm methods often fall short of capturing the nuanced dependencies among tasks, data, and resources in large-scale clouds.

Graph Neural Networks (GNNs) address this limitation by learning rich representations over graph-structured data [13], propagating information along edges to model complex relationships [14]. In cloud computing, GNNs have been employed for energy optimization, resource allocation, VM migration, and load balancing [15]. Representing tasks, resources, or VMs as graph nodes and their interactions as edges allows a GNN to reason over the full system context [16], making it well suited to large, dynamic environments.

Policy-Gradient (PG) reinforcement-learning methods complement GNNs by directly optimizing a parameterized policy to maximize expected long-term rewards [17]. In cloud resource management, PG algorithms can learn adaptive allocation strategies responsive to current system states and task requirements [18]. Combining GNN representation learning with PG decision making thus yields a powerful optimization paradigm that considers both fine-grained structural dependencies and long-horizon objectives.

A key practical complication is the intrinsic tension between minimizing energy consumption and maximizing performance. Prioritizing one objective risk SLA violations or excessive operational costs [19]. A multi-objective perspective is therefore essential: only by balancing competing goals can we satisfy the diverse demands of big-data workloads.

This paper introduces a unified framework that fuses GNN modeling with a bespoke PG optimizer under a composite reward that explicitly balances energy and performance. The resulting policies capture task–resource intricacies, adapt to evolving workloads, and sustain energy-aware yet high-performing operation. To our knowledge, this is the first study to integrate GNNs, PG algorithms, and multi-objective optimization specifically for energy-efficient big-data processing in cloud environments.

II. Methods

This research undertakes the design and exhaustive evaluation of a new multi-objective optimization framework that targets energy-efficient big-data processing in cloud environments. The framework's overarching goal is to overcome the shortcomings of current optimizers by (i) accurately capturing the fine-grained dependencies that bind tasks to resources and (ii) striking a deliberate balance between energy consumption and overall system performance.

To fulfil this goal, we follow a quantitative experimental methodology encompassing both the meticulous construction of the framework and its rigorous assessment through large-scale simulations. The implementation combines three key components: (1) a GNN module for expressive task–resource representation, (2) an RL-based policy-gradient engine that adapts allocation decisions online, and (3) a unifying multi-objective scheme that reconciles energy use with performance targets.

Empirical studies are carried out in a purpose-built simulation environment based on CloudSim, an extensible open-source toolkit that faithfully mirrors real data-centre behaviors. Multiple heterogeneous servers—differing in processing power, memory capacity, and energy profiles—are instantiated to reproduce realistic resource dynamics and their impact on efficiency.

Workloads are synthesized to emulate true big-data jobs, with diverse execution times, memory footprints, deadlines, and inter-task dependencies. To gauge scalability and robustness, we employ both controlled synthetic traces and the Google Cluster Trace, the latter offering production-grade workload characteristics.

The proposed framework is benchmarked against three strong baselines: a Greedy Heuristic (GH), a Multi-Objective Genetic Algorithm (MOGA), and a Deep Reinforcement Learning (DRL) scheduler. Performance is quantified with respect to total energy usage, makespan, mean task completion time, allocation fairness, and Pareto optimality. Throughout the simulation runs we collect detailed logs, apply appropriate statistical analyses, and present results via intuitive visualizations.

Because experiments are performed under fully controlled, reproducible conditions rather than statistical sampling, traditional power-analysis calculations are unnecessary. The study instead centers on modelling fidelity and comprehensive performance appraisal within a simulated cloud that mirrors real-world settings.

III. Architecture of Big Data Services in the Cloud

Effective ingestion, storage, and analysis of massive data volumes rely on sophisticated, multilayered cloud architectures expressly crafted to exploit the scalability, elasticity, and cost advantages of the cloud paradigm [20]. As illustrated in **Fig. 1**, a canonical architecture comprises several tiers. At the foundation lies the physical-infrastructure layer—an extensive mesh of servers, storage arrays, and networking hardware. Above this, the virtualization layer abstracts those physical resources, enabling rapid deployment of isolated virtual machines or containers and thereby fostering high resource utilization and true multi-tenancy [21].

Above the virtualization substrate, a series of data-oriented tiers orchestrates the entire big-data life-cycle. First, a distributed-storage layer delivers fault-tolerant, high-throughput persistence for structured, semi-structured, and unstructured datasets, typically relying on file systems such as Hadoop HDFS or Ceph [22]. Next, a distributed-processing layer enables large-scale parallel computation through engines like Apache Hadoop, Spark, or Flink [23]. To cope with workload volatility, an intelligent resource-management tier continuously adjusts CPU, memory, and network allocations, while a service-provisioning layer automates deployment, configuration, and maintenance of data-analytics applications.

Yet the very attributes that make these architectures powerful—elastic resource pools and on-demand scaling—also complicate energy stewardship. Highly dynamic workloads, intricate task-data-resource interdependencies, and heterogeneous hardware all conspire to make energy optimization remarkably challenging [24]. Moreover, operators must juggle conflicting goals, balancing performance against power savings within sprawling, multi-tenant data centers.

Overcoming these obstacles calls for holistic, adaptive optimization frameworks that capture workload dynamics and architectural complexity, then make intelligent decisions to harmonies energy consumption with performance. The rest of this paper introduces such a framework, built on GNN modelling and reinforcement-learning (RL) techniques, to meet the demanding requirements of energy-efficient big-data processing in modern cloud environments.

IV. Related Works

A. Energy-Efficient Big Data Processing in Cloud Computing

The dramatic rise of big-data workloads and the corresponding surge in cloud-data-centre energy bills have made power-aware processing a priority for both academia and industry. Numerous optimization strategies and frameworks have therefore been proposed to curb energy use while preserving performance and meeting QoS targets.

Early efforts relied on heuristic rules crafted from domain expertise. Mashayekhy et al. developed an energy-aware scheduler for MapReduce that favors the most power-efficient servers when assigning tasks, explicitly accounting for resource heterogeneity [25]. Likewise, Zhao et al. designed a heuristic data-placement algorithm that leverages access-pattern knowledge and device-power profiles to cut storage energy overheads [26].

More recently, data-driven techniques have gained traction. Berral et al. introduced a machine-learning framework in which neural models predict future resource demand and provision capacity accordingly, reducing wasted energy in cloud data centres [27]. Mostafavi et al. advanced this line of work by formulating task scheduling as a reinforcement-learning (RL) problem; their agent continually refines its policy to minimize energy while honoring performance constraints [28].

To reconcile conflicting objectives such as energy and throughput, researchers have turned to multi-objective optimization. Wang et al. framed task scheduling as a bi-objective problem (energy and makespan) and applied a genetic algorithm to approximate Pareto-optimal schedules [29]. Shrimali et al. employed particle-swarm optimization to balance power draw against QoS guarantees during resource allocation [30].

Despite these advances, important gaps remain. Many solutions concentrate on isolated facets—task scheduling or resource placement—without modelling the intricate coupling among tasks, data, and compute nodes. Furthermore, most adopt static or offline optimization, assuming workload characteristics are known a priori. Real clouds face stochastic, rapidly changing demands that require online, adaptive control. Finally, striking a satisfactory compromise between energy and performance continues to be difficult because the objectives are inherently at odds [31]. Addressing these shortcomings calls for comprehensive frameworks capable of real-time learning and holistic decision-making—an issue taken up in the next subsection.

B. GNNs and Reinforcement Learning for Optimization

GNNs and RL have emerged as potent tools for complex optimization across many domains, including cloud resource management. GNNs excel at encoding the rich relational structure among entities, while RL provides a principled way to learn decision policies through environmental interaction.

In cloud contexts, GNNs have been used to capture the multifaceted links among servers, tasks, and network paths. Shabka et al. modelled a data centre as a graph whose nodes are servers and edges are links; the learned representations guided energy-aware resource allocation and improved overall efficiency [32]. Chen et al. extended this idea to edge environments, encoding task dependencies and device constraints as a graph and using a GNN to derive schedules that shorten completion time and cut energy consumption [33].

RL, meanwhile, has proven effective for dynamic resource control. Liu et al. adopted a Deep Q-Network (DQN) to learn allocation policies that boost throughput while trimming power use [34]. Kang et al. proposed a hierarchical RL scheme with separate agents for inter- and intra-data-centre scheduling, collectively driving down total energy consumption [35].

Integrating GNN representation learning with RL decision-making further enhances optimization power. Thein et al. demonstrated this synergy by embedding task-resource relationships with neural networks and letting an RL agent exploit those embeddings to select high-quality allocations, surpassing heuristics and standalone ML in both energy savings and QoS compliance [36].

Nonetheless, several research challenges persist. Scaling GNN and RL methods to clouds comprising thousands of servers and millions of tasks demands efficient, distributed training algorithms. Policy transparency is another concern: as models grow more complex, improving interpretability becomes vital for operator trust. Finally, fusing GNN-RL frameworks with complementary paradigms—such as multi-objective evolutionary search or game-theoretic coordination—could yield even more resilient, adaptive solutions for energy-efficient big-data processing in dynamic cloud settings.

V. Multi-Objective Optimization Framework

A. Overview

This study proposes a multi-objective optimization framework that targets energy-efficient big-data processing in cloud environments while simultaneously maximizing system performance and respecting QoS requirements. By uniting GNNs with Reinforcement Learning (RL), the framework overcomes key shortcomings of earlier methods and offers a holistic solution for cloud-scale analytics.

The architecture comprises three synergistic modules.

1) GNN-based modelling: Features describing tasks and resources are encoded into a task–resource graph that reflects the fine-grained dependencies and constraints inherent to the cloud. The GNN transforms this graph into compact, low-dimensional embeddings for both tasks and resources, serving as rich inputs for downstream decision making.

2) RL-based optimization: Using the embeddings, an RL agent learns resource-allocation and scheduling policies

through continual interaction with the simulated cloud. The agent optimizes a reward that jointly considers energy, throughput, and service quality, thereby navigating the inevitable trade-offs among these objectives.

3) Multi-objective strategy: A flexible weighting mechanism shapes the composite reward, allowing operators to priorities energy savings, performance, or QoS as needed.

Figure 2 illustrates the overall workflow. The GNN captures intricate task-resource interactions that heuristics or flat neural models often miss, yielding more informed decisions. The RL component continuously refines its policy in response to workload volatility and resource heterogeneity. The multi-objective formulation lets users explicitly tune the balance among competing goals. By learning directly from graph-structured data and operating online, the framework scales to large, heterogeneous clouds and diverse workload profiles, outperforming heuristics and single-objective methods in both energy efficiency and overall performance.

B. Problem Formulation

In this section, we formally define the energy-efficient big data processing problem within cloud computing environments. We introduce the necessary mathematical notations and symbols to model the system accurately and present the corresponding optimization objectives alongside the associated constraints.

Given a cloud computing environment with a set of M servers denoted as $S = s_1, s_2, \dots, s_M$, and a set of N big data tasks denoted as $T = t_1, t_2, \dots, t_N$, the goal is to find an optimal allocation of tasks to servers and determine the processing order of tasks on each server, such that the overall energy consumption is minimized while the system performance is maximized and the QoS requirements are satisfied. We define the following notations. x_{ij}, a binary variable indicating whether task t_i is assigned to server $s_j(x_{ij}= 1)$ or not $(x_{ij}= 0)$. y_{ijk}, a binary variable indicating whether task t_i is processed before task t_k on server $s_j(y_{ijk} = 1)$ or not $(y_{ijk}= 0)$. p_i is the processing time of task t_i. e_j is the energy consumption per unit time of server s_j. c_j the processing capacity of server s_j. the deadline of task t_i. m_i, the memory requirement of task t_i. b_j the available memory of server the available memory of server s_j.

The optimization objectives can be formally expressed as follows. The first objective is to minimize the total energy consumption of the cloud computing system:

$$\text{minimize} \sum_{j=1}^{M} e_j \sum_{i=1}^{N} p_i x_{ij} \tag{1}$$

where e_j represents the energy consumption per unit time of server s_j, p_i denotes the processing time of task t_i, and x_{ij} is a binary variable indicating the assignment of task t_i to server s_j.

Maximizing the system throughput, which is defined as the number of tasks completed per unit of time:

$$\text{maximize} \frac{\sum_{i=1}^{N} \sum_{j=1}^{M} x_{ij}}{T} \tag{2}$$

where T represents the total processing time.

The optimization problem is subject to the following constraints. Each task must be assigned to one and only one server:

$$\sum_{j=1}^{M} x_{ij} = 1, \forall i \in 1,2,\dots,N \tag{3}$$

This constraint guarantees that each task is handled by a single server. Additionally, the processing capacity of each server must not be exceeded:

$$\sum_{i=1}^{N} p_i x_{ij} \le c_j, \forall j \in 1,2,\dots,M \tag{4}$$

where c_j denotes the processing capacity of server s_j. The deadline of each task must be satisfied in **equation (5)**, where d_i represents the deadline of task t_i, and y_{ijk} is a binary variable indicating the processing order of tasks t_i and t_k on server s_j.

The memory requirement of each task must not exceed the available memory of the assigned server:

$$m_i x_{ij} \le b_j, \forall i \in 1,2,\dots,N, \forall j \in 1,2,\dots,M \tag{6}$$

where $m_i$denotes the memory requirement of task t_i, and b_j represents the available memory of server s_j.

The processing order of tasks on each server must be consistent, as shown in **equation (7)**.

This constraint ensures that the execution order of tasks on each server is clearly defined and consistent. The resulting optimization problem is formulated as a multi-objective mixed-integer linear programming (MILP) problem, which is known to be NP-hard. To address this complexity effectively, we propose a novel approach that integrates GNN-based task and resource modeling, RL-based optimization, and a tailored multi-objective optimization strategy. The details of this approach are presented in the subsequent sections.

C. GNN-based Task and Resource Modelling

To effectively capture the complex dependencies and interactions among tasks and resources in cloud computing environments, we propose a GNN-based approach for modeling and representation learning. GNNs have demonstrated exceptional capability in extracting meaningful patterns from graph-structured data, making them particularly well-suited to the challenges of our problem setting.

We define a task-resource graph $G = (V, E)$, where V represents the set of nodes and E denotes the set of edges. The node set V consists of two types of nodes: task nodes $V_T = t_1, t_2, \dots, t_N$ and resource nodes $V_R = r_1, r_2, \dots, r_M$, where N is the number of tasks and M is the number of resources (servers). Each task node $t_i \in V_T$ is associated with a feature vector $x_{t_i} \in R^{d_t}$, which encodes the characteristics of the task, such as its processing time, memory requirement, and deadline. Similarly, each resource node $r_j \in V_R$ is associated with a feature vector $x_{r_j} \in R^{d_r}$, which captures the properties of the resource, such as its processing capacity, available memory, and energy consumption per unit time.

The edges in the task-resource graph represent the compatibility and constraints between tasks and resources. We define two types of edges: task-to-resource edges $E_{TR} \subseteq V_T \times V_R$ and task-to-task edges $E_{TT} \subseteq V_T \times V_T$. A task-to-resource edge $(t_i, r_j) \in E_{TR}$ indicates that task t_i can be assigned to the resource r_j, subject to the resource's capacity and the task's requirements. A task-to-task edge $(t_i, t_k) \in E_{TT}$ represents the dependencies or communication requirements between tasks t_i and t_k.

Given the task-resource graph G, our goal is to learn a low-dimensional representation (embedding) for each node, which captures the structural and semantic information of the graph. We employ a GNN model to achieve this goal. The GNN takes the initial node features $X = x_{t_1}, \dots, x_{t_N}, x_{r_1}, \dots, x_{r_M}$ as input and performs multiple layers of message passing and aggregation to update the node embeddings. At each layer l of the GNN, the embedding of a node v is updated by aggregating the embeddings of its neighbors $\mathcal{N}_{(v)}$ and combining them with its own embedding from the previous layer, as shown in **equation (8)**, where $h_v^{(l)} \in R^{d^{(l)}}$ is the embedding of node v at layer l, $W^{(l)} \in R^{d^{(l)} \times d^{(l-1)}}$ and $b^{(l)} \in R^{d^{(l)}}$ are learnable weight matrix and bias vector, respectively, $\mathrm{AGG}(\cdot)$ is an aggregation function (e.g., mean, max, or sum) that combines the embeddings of the neighboring nodes, and $\sigma(\cdot)$ is a non-linear activation function (e.g., ReLU or sigmoid).

The GNN is trained using a supervised learning approach, where the objective is to minimize a loss function that quantifies the discrepancy between the predicted assignments-based on the learned node embeddings—and the ground-truth assignments. The training process consists of forward propagation to compute the node embeddings, followed by backpropagation to update the model parameters (weights and biases) using the gradients of the loss function. Once trained, the GNN model is capable of generating informative embeddings for both tasks and resources, effectively capturing their intrinsic properties and interdependencies. These embeddings are then fed into the subsequent optimization module, facilitating more effective and efficient decision-making for task allocation and resource management in the cloud computing environment.

D. Reinforcement Learning-based Optimization

Building upon the GNN-based task and resource modeling, we propose an RL-based approach to optimize the allocation of tasks to resources within the cloud computing environment. RL offers a robust framework for sequential decision-making under uncertainty, enabling the learning of optimal policies through interaction with the environment. We formulate the optimization problem as a Markov Decision Process (MDP), where the RL agent interacts with the cloud computing environment to learn the optimal task allocation policy. The key elements of the MDP are defined as follows.

The state s_t at time step t is represented by the embeddings of tasks and resources generated by the GNN model, along with additional global features such as the overall system throughput and energy consumption. Formally, $s_t = (h_{t_1}, \dots, h_{t_N}, h_{r_1}, \dots, h_{r_M}, g_t)$, where h_{t_i} and h_{r_j} are the embeddings of task t_i and resource r_j, respectively, and g_t represents the global features. The action a_t at time step t corresponds to the allocation of tasks to resources. We define the action space as a discrete set of possible task-to-resource assignments, where each action $a_t = (a_{t,1}, \dots, a_{t,N})$ is a vector of resource indices, with $a_{t,i} \in 1, \dots, M$ indicating the resource to which task t_i is assigned. The reward function $r(s_t, a_t)$ measures the immediate performance of the system after taking action a_t in state s_t. We design the reward function to align with the optimization objectives, considering both the energy consumption and the system throughput. Specifically, $r(s_t, a_t) = -\lambda_1 \cdot \mathrm{energy}(s_t, a_t) + \lambda_2 \cdot \mathrm{throughput}(s_t, a_t)$, where $\mathrm{energy}(s_t, a_t)$ and $\mathrm{throughput}(s_t, a_t)$ are the total energy consumption and system throughput achieved by taking action a_t in state s_t, and λ_1 and λ_2 are positive weights that balance the two objectives.

The goal of the RL agent is to learn a policy $\pi(a_t|s_t)$ that maximizes the expected cumulative reward over a horizon of T time steps:

$$J(\pi) = \mathbb{E}_\pi[\sum t = 0^{T-1} \gamma^t \cdot r(s_t, a_t)] \quad (9)$$

where $\gamma \in [0,1]$ is a discount factor that determines the importance of future rewards.

To solve the optimization problem, we employ a variant of the DQN algorithm, which combines Q-learning with deep neural networks for function approximation. The DQN agent maintains an action-value function $Q(s, a; \theta)$, parameterized by a neural network with weights θ, which estimates the expected cumulative reward of taking action a in state s. The agent interacts with the environment by selecting actions based on an ϵ-greedy policy, where it chooses the action with the highest Q-value with probability $1 - \epsilon$ and a random action with probability ϵ for exploration. The DQN is trained using experience replay and target network techniques to stabilize the learning process. The agent stores its experiences (s_t, a_t, r_t, s_{t+1}) in a replay buffer and periodically samples a batch of experiences to update the Q-network parameters θ by minimizing the temporal difference error, as shown in **equation (10)**, where D is the replay buffer and θ^- are the parameters of a target Q-network that is periodically updated with the current Q-network parameters.

The integration of GNN-based modeling with RL-based optimization offers several key advantages. The GNN embeddings serve as compact and informative representations of tasks and resources, effectively capturing their intrinsic characteristics and interdependencies. These embeddings enable the RL agent to make more informed and efficient allocation decisions, while dynamically adapting to the stochastic and evolving nature of the cloud computing environment. Furthermore, the use of the DQN algorithm enhances sample efficiency and training stability, making it particularly suitable for the proposed optimization problem. By combining GNN-based modeling with RL-based optimization, our approach is capable of addressing the complexity and scale of modern cloud computing systems, learning optimal policies that promote both energy efficiency and high performance in big data processing. These learned policies can adapt to fluctuating workloads and changing system conditions, offering a flexible and robust solution for resource management in the cloud.

E. Multi-Objective Optimization Strategy

In the context of energy-efficient big data processing within cloud computing environments, optimizing for a single objective-such as minimizing energy consumption or maximizing system throughput-is often insufficient. In practice, these objectives frequently conflict, necessitating trade-offs to achieve a balanced and satisfactory outcome. To address this challenge, we propose a multi-objective optimization strategy that integrates seamlessly with our GNN-based modeling and RL-based optimization framework.

Central to our multi-objective optimization strategy is the design of a composite reward function that incorporates multiple optimization objectives. Let $O = o_1, o_2, \dots, o_K$ be the set of K objectives we consider, such as energy consumption, system throughput, resource utilization, and response time. For each objective o_k, we define a corresponding reward function $r_k(s_t, a_t)$ that measures the performance of the system concerning that objective when taking action a_t in state s_t. To combine these individual reward functions into a single composite reward, we introduce a set of weight parameters $w = (w_1, w_2, \dots, w_K)$, where $w_k \geq 0$ and $\sum_{k=1}^{K} w_k = 1$. The composite reward function is then defined as a weighted sum of the individual rewards:

$$r(s_t, a_t) = \sum_{k=1}^{K} w_k \cdot r_k(s_t, a_t) \qquad (11)$$

The weight parameters $\boldsymbol{w}$ represent the relative importance of each objective and can be adjusted based on user preferences or system requirements. By setting appropriate weights, we can prioritize certain objectives over others and control the trade-off between conflicting goals. However, optimizing for multiple objectives simultaneously presents several challenges. First, the objectives may have different scales and ranges, making it difficult to compare and aggregate them directly. To address this issue, we normalize the individual reward functions using min-max scaling:

$$\hat{r}k(s_t, a_t) = \frac{r_k(s_t, a_t) - min a r_k(s_t, a)}{\underset{a}{max} r_k(s_t, a) - \underset{a}{min} r_k(s_t, a)} \qquad (12)$$

where $\min_a r_k(s_t, a)$ and $\max_a r_k(s_t, a)$ are the minimum and maximum rewards attainable for objective o_k in state s_t, respectively. This normalization ensures that all rewards are in the range [0, 1], facilitating their aggregation in the composite reward function.

Another challenge lies in the potentially large and complex search space of the multi-objective optimization problem. To efficiently explore this space and find Pareto-optimal solutions, we employ a multi-objective variant of the DQN algorithm, called Multi-Objective DQN (MO-DQN). MO-DQN maintains a separate Q-network for each objective, denoted as $Q_k(s, a; \theta_k)$, and learns to approximate the optimal action-value function for each objective simultaneously. The training procedure for MO-DQN follows a similar approach to the standard DQN, but with a modified loss function that considers the composite reward, just refer to **equation (13)**, where $\hat{r}_{k,t}$ is the normalized reward for objective o_k at time step t.

During the optimization process, the MO-DQN agent learns to balance multiple objectives based on the specified weights. By exploring different weight configurations, we obtain a set of Pareto-optimal solutions that represent the best trade-offs between the conflicting objectives. Decision-makers can then select the solution that most closely aligns with their preferences and system constraints.

F. Algorithm Design and Implementation

In this section, we present the complete algorithmic flow of our proposed multi-objective optimization framework for energy-efficient big data processing in cloud computing environments. We discuss the input, output, and main steps of the algorithm, along with its time and space complexity, scalability, and parallelization aspects. Key code snippets and implementation details are provided to facilitate understanding and reproducibility. **Algorithm 1** outlines the main steps of our optimization framework. The input to the algorithm includes the task set $\mathcal{T}$, the resource set $\mathcal{R}$, the objective set $\mathcal{O}$, and the weight vector w. The output is a set of Pareto-optimal task allocation policies Π^* that balance the multiple objectives based on the given weights.

The algorithm begins by constructing the task-resource graph G based on the given task set $\mathcal{T}$and resource set $\mathcal{R}$ (line 1). The GNN model parameters θ are initialized (line 2). The main optimization process is performed over E episodes (lines 3-18). In each episode, the MO-DQN agent is initialized with K Q-networks corresponding to the K objectives (line 4).

Within each episode, the algorithm proceeds for S steps (lines 5-16). At each step, the GNN model generates embeddings for tasks and resources based on the current state of the task-resource graph (lines 6-7). The current state s_t is formed by concatenating the task embeddings, resource embeddings, and global features (line 8). The MO-DQN agent selects an action a_t based on its exploration strategy (line 9), executes the action, and observes the rewards $r_1, r_2, \dots, r_K$ and the next state s_{t+1}(line 10). The individual rewards are normalized using min-max scaling (line 11), and the composite reward r_t is calculated as the weighted sum of the normalized rewards (line 12). The experience tuple (s_t, a_t, r_t, s_{t+1}) is stored in the replay buffer D (line 13). A batch of experiences is sampled from D, and the Q-networks are updated using the loss function $L(\theta_1, \dots, \theta_K)$ (line 14). The GNN model parameters θ are also updated based on the sampled experiences (line 15). After each episode, the learned task allocation policy π is evaluated, and if it is Pareto-optimal, it is added to the set of optimal policies Π (line 17). Finally, the algorithm returns the set of Pareto-optimal task allocation policies Π^* (line 19).

The time complexity of the algorithm is determined by the number of episodes E, the number of steps S per episode, and the complexity of the GNN and MO-DQN models. Let C_{GNN} and $C_{\text{MO-DQN}}$ denote the time complexity of the GNN and MO-DQN models, respectively. The overall time complexity is $O\left(E \cdot S \cdot \left(C_{\text{GNN}} + C_{\text{MO-DQN}}\right)\right)$. The space complexity is dominated by the size of the replay buffer D and the memory required to store the GNN and MO-DQN models.

To enhance the scalability and efficiency of the algorithm, several optimization techniques can be applied. Parallel computing can accelerate the generation of task and resource embeddings using the GNN model. The experience replay buffer can be designed with efficient data structures and sampling strategies to accommodate large-scale datasets. The Q-networks can be parameterized using lightweight neural network architectures to reduce both memory usage and computational overhead. The algorithm was implemented using deep learning frameworks such as PyTorch, which provide efficient primitives for constructing and training GNN and RL models. Additionally, the task-resource graph was represented using graph libraries like NetworkX, offering intuitive APIs for graph manipulation and computation.

VI. Experimental Implementation and Results

A. Experimental Setup

To evaluate our framework, we conducted extensive experiments in a large-scale cloud computing simulation environment. The simulator was developed using CloudSim [37], a widely adopted open-source cloud simulation toolkit. We extended CloudSim to support the modeling of energy consumption, resource heterogeneity, and task dependencies. The experiments were carried out on a machine equipped with an Intel Xeon E5-2680 v4 CPU (2.40 GHz, 28 cores) and 128 GB of RAM, running Ubuntu 18.04 LTS. The GNN model and the MO-DQN agent were implemented using PyTorch [38] and PyTorch Geometric [39].

B. Datasets

To comprehensively evaluate the performance and effectiveness of our proposed multi-objective optimization framework, we utilize both synthetic and real-world datasets. The synthetic datasets, generated using the CloudSim toolkit [37], offer a controlled environment for analyzing the framework's behavior under various conditions. By adjusting parameters such as the number of tasks, resource characteristics, and task dependencies, we examine the framework's scalability and adaptability to diverse workload patterns and resource configurations. We construct three synthetic datasets with varying scales and complexity levels:

(1) Medium-scale dataset: This dataset comprises 10,000 tasks and 500 resources, representing a more complex and resource-intensive cloud computing environment. The tasks feature diverse execution times, resource demands, and inter-task dependencies. This setup allows us to evaluate the framework's capability to manage larger workloads and optimize resource allocation under increased complexity.

(2) Large-scale dataset: To further challenge the framework's scalability, we generate a large-scale dataset with 100,000 tasks and 5,000 resources. This dataset simulates a highly complex and resource-constrained cloud computing scenario, with tasks exhibiting intricate dependencies and resource requirements. Evaluating the framework on this dataset demonstrates its ability to handle massive-scale problems and make effective optimization decisions.

While synthetic datasets provide valuable insights under controlled scenarios, it is also essential to validate the framework's practicality using real-world data. For this purpose, we employ a widely used and publicly available dataset:

Google Cluster Trace [40]: This dataset contains rich information about workload traces from Google's production cluster, including task resource requirements, durations, and dependencies. To align with the scale of our experimental setup, we preprocess the dataset and extract a representative subset comprising 50,000 tasks and 1,000 resources. This subset retains the essential characteristics and patterns of the original data while reducing computational complexity. Evaluating the framework on the Google Cluster Trace enables us to assess its real-world performance and applicability in production environments.

By leveraging both synthetic and real-world datasets, we aim to provide a comprehensive and rigorous evaluation of our proposed framework. The synthetic datasets facilitate an in-depth analysis of scalability, adaptability, and performance under controlled conditions, while the real-world dataset validates the framework's effectiveness in handling complex, large-scale workloads reflective of practical cloud computing scenarios.

C. Evaluation Metrics

We evaluated the performance of our framework using the following metrics:

(1) Energy Consumption (EC): The total energy consumed by the cloud computing system to process the given workload. This metric is computed based on resource utilization and the power consumption characteristics of the underlying resources.

(2) Makespan (MS): The total time required to complete all tasks in the workload. This metric reflects the overall efficiency and performance of the task allocation and scheduling process.

(3) Average Task Completion Time (ATCT): The average time taken to complete an individual task within the workload. This metric offers insights into system responsiveness and user experience.

(4) Fairness: The degree to which resources are allocated equitably among tasks. We employ Jain's Fairness Index [41] to quantitatively assess the fairness of resource distribution.

(5) Pareto Optimality (PO): The capability of the framework to discover Pareto-optimal solutions that effectively balance energy consumption and performance. We use the hypervolume indicator [42] to measure the quality of the Pareto front obtained.

D. Experimental Results and Analysis

We compared our proposed framework with three state-of-the-art methods for energy-efficient big data processing in cloud computing environments: GH [43], a simple heuristic that allocates tasks to the most energy-efficient resources based on current utilization; MOGA [44], a genetic algorithm-based approach optimizing task allocation with respect to energy consumption and makespan; and DRL [45], a DRL method that learns task allocation policies aimed at minimizing energy consumption while satisfying performance constraints. We conducted experiments on both synthetic and real-world datasets, evaluating the performance of each method using defined metrics. The results are presented and analyzed in the following subsections.

Energy Consumption

Figure 3 illustrates the comparative energy consumption of our proposed framework against the GH, MOGA, and DRL methods across the different evaluated datasets. On the medium-scale synthetic dataset, our framework demonstrated significant energy savings, reducing energy consumption by 40.0% compared to GH, by 27.6% compared to MOGA, and by 17.1% compared to DRL, highlighting its effectiveness even in moderately complex environments. The advantages of our framework were particularly pronounced on the large-scale synthetic dataset, where it successfully reduced energy consumption by 35.0% when benchmarked against GH, 27.8% against MOGA, and 18.8% against DRL. These results underscore the framework's capability to maintain high energy efficiency when managing extensive and complex workloads. Furthermore, the practical applicability and robust performance of our framework were confirmed on the Google Cluster Trace (real-world) dataset. Here, it achieved substantial energy savings, with reductions of approximately 39.2% compared to GH, 34.8% compared to MOGA, and 30.8% compared to DRL. These specific improvements fall within the overall observed energy savings range of 31% to 39% against these contemporary methods, validating its performance with real-world operational data.

Across all tested datasets, our proposed framework consistently achieved the lowest energy consumption. This superior performance is primarily attributed to its advanced capability to accurately model complex task-resource dependencies using the GNN model. Furthermore, the MO-DQN agent effectively learns and implements energy-efficient task allocation policies. The integrated multi-objective optimization strategy also plays a crucial role by enabling the framework to skillfully balance energy consumption with other critical performance metrics, thereby resulting in significantly enhanced overall operational efficiency.

Makespan and Average Task Completion Time

Figures 4(a) and **4(b)** illustrate the comparative performance trends for makespan and average task completion time (ATCT), respectively, across varying task loads (from 20,000 to 100,000 tasks) on the large-scale synthetic dataset. Our proposed framework consistently demonstrates strong performance against the benchmark methods across both metrics as the workload intensity increases.

As shown in Figure 4(a), for makespan on the large-scale synthetic dataset, our framework achieves significant reductions. At the maximum evaluated task load of 100,000 tasks, it reduces makespan by 28.0% compared to the GH and 17.0% compared to the MOGA. Furthermore, our framework performs slightly better than the DRL method, achieving a makespan of 720-time units compared to DRL's 725-time units under the same conditions. The plotted trends clearly show our framework maintaining this competitive makespan advantage across the different workload intensities evaluated.

Figure 4(b) details the ATCT on the same large-scale synthetic dataset. Our framework consistently yields the lowest average task completion times across all tested task loads. Specifically, at the 100,000.00 task load, our proposed method reduces ATCT by 30.0% relative to GH, 18.6% relative to MOGA, and 2.8% relative to DRL. The trend lines in Figure 4(b) further highlight the ability of our framework to manage tasks more efficiently, leading to quicker average completion times even as the system load escalates from 20,000 to 100,000 tasks. While these figures focus on the large-scale synthetic dataset, it's noteworthy that on the real-world datasets (as detailed elsewhere in our results), our framework also achieves makespan reductions ranging from 23% to 35%, along with ATCT improvements of 19% to 31%.

These findings highlight the effectiveness of our framework in balancing the trade-off between energy consumption and performance. The GNN-based model captures intricate dependencies among tasks and resources, allowing the framework to make informed and adaptive task allocation decisions. This leads to optimized outcomes in both energy efficiency and execution performance.

Fairness and Pareto Optimality

Figure 5 presents the fairness comparison, utilizing Jain's Fairness Index, as a function of increasing task load (from 20,000 to 100,000 tasks) on the large-scale synthetic dataset. The trend lines clearly demonstrate that our proposed framework consistently achieves the highest fairness scores across all evaluated workload intensities. Specifically, our framework maintains a Jain's Fairness Index between 0.92 and 0.96, showcasing remarkable stability and equitable resource distribution even as the system load escalates. In contrast, the GH method exhibits the lowest fairness, declining from 0.65 to 0.52, while the MOGA performs better than GH but still shows a decrease in fairness from 0.75 to 0.66 under increasing load. The DRL method maintains a relatively good fairness score, fluctuating between 0.84 and 0.87, but remains consistently below our proposed framework. While this figure focuses on the large-scale synthetic dataset, the superior fairness of our approach is a consistent finding across all datasets evaluated, highlighting its effectiveness in distributing resources equitably. The multi-objective optimization strategy integrated into our framework plays a key role in preventing any task from being deprived of resources, while simultaneously optimizing for both energy efficiency and performance.

Figure 6 depicts the hypervolume comparison of Pareto-optimal solutions. Our framework attains the highest hypervolume across all datasets, showcasing its superior capability in balancing the trade-offs between energy consumption and system performance. The elevated hypervolume values reflect the robustness of our multi-objective optimization approach in thoroughly exploring the solution space and identifying Pareto-optimal solutions that surpass those derived from benchmark methods. Figure 6 depicts the hypervolume comparison of Pareto-optimal solutions across seven distinct datasets: S-Syn (Small-Synthetic), M-Syn (Medium-Synthetic), L-Syn (Large-Synthetic), RW-Net (RealWorld-NetworkBound), RW-CPU (RealWorld-CPUBound), RW-Mem (RealWorld-MemoryBound), and Mix-Lrg (Mixed-LargeScale). Our framework consistently attains the highest hypervolume values, ranging from approximately 0.88 to 0.94 across these datasets, showcasing its superior capability in balancing the trade-offs between energy consumption and system performance. Among the baseline methods, DRL performs as the closest competitor, achieving hypervolume indicators generally between 0.80 and 0.88, and exhibiting particular strength on datasets such as RW-CPU and Mix-Lrg. MOGA typically ranks third, with its hypervolume values fluctuating between approximately 0.70 and 0.78, showing varied responsiveness to dataset characteristics, for instance, with improved performance on L-Syn and Mix-Lrg. The GH method consistently yields the lowest hypervolume indicators (ranging from 0.62 to 0.68), though it shows relative stability under RW-Net and RW-Mem conditions. The elevated and consistent hypervolume achieved by our framework reflects the robustness of our multi-objective optimization approach in thoroughly exploring the solution space and identifying Pareto-optimal solutions that significantly surpass those derived from the benchmark techniques.

Scalability Analysis Under Varying Workload Intensities

To further evaluate our framework's performance with respect to makespan, we conducted a comprehensive scalability analysis under varying workload intensities and task dependency complexities. We systematically increased the workload intensity from 20% to 100% of maximum capacity while measuring the impact on makespan across all four methods (our GNN-based framework, GH, MOGA, and DRL).

Figure 7 presents the normalized makespan as a function of workload intensity, ranging from 20% to 100%, for our framework and the baseline methods GH, MOGA, and DRL. A lower normalized makespan signifies better performance, with a value of 1.0 representing the baseline performance at 20% workload. Our framework demonstrates superior scalability, as its normalized makespan shows the least increase with rising workload intensity. Specifically, at 100% workload intensity, our framework's normalized makespan reaches 1.11, indicating only an 11% performance degradation from its 20% baseline. In contrast, the DRL, MOGA, and GH methods experience more substantial degradations under full workload, with their normalized makespans increasing to 1.18 (18% degradation), 1.23 (23% degradation), and 1.27 (27% degradation), respectively. This clearly illustrates our framework's capability to maintain higher operational efficiency and more consistent performance as the system load escalates.

We also analyzed the effect of task dependency complexity on makespan performance. **Figure 8** illustrates how makespan varies across three levels of task dependency complexity (low, medium, and high) for each method. Our GNN-based approach exhibits the least sensitivity to increasing task complexity due to its ability to effectively model and adapt to complex task-resource dependencies.

Performance Under Resource Constraints

To evaluate how different resource constraint scenarios affect the Average Task Completion Time (ATCT), we conducted experiments under four distinct resource limitation conditions: CPU-constrained, memory-constrained, network-constrained, and balanced resources. This experiment is particularly relevant for real-world cloud deployments where specific resource types may become bottlenecks.

Figure 9 presents the ATCT for all four methods under these different resource constraint scenarios. Our framework consistently outperforms the comparison methods across all scenarios, with particularly significant improvements under memory-constrained conditions, which are common in big data environments. Specifically, our approach achieves 37-42% reduction in ATCT compared to GH, 29-33% compared to MOGA, and 15-21% compared to DRL in memory-constrained environments.

We further analyzed the resource utilization patterns for each method under these constraint scenarios. **Figure 10** shows the resource utilization efficiency (RUE) metric, defined as the ratio of useful work performed to total resources consumed. Our framework achieves the highest RUE across all scenarios, demonstrating its ability to intelligently allocate resources even under severe constraints.

To further probe the robustness of our framework under resource scarcity, **Figure 11** illustrates the impact of varying degrees of memory constraint on Resource Utilization Efficiency (RUE). The x-axis represents decreasing memory availability, from 100% (no specific constraint beyond baseline) down to 20% (severe memory limitation). Our proposed framework consistently maintains the highest RUE across all levels of memory availability. Notably, as memory becomes increasingly scarce (moving from left to right on the graph), our framework exhibits a more graceful degradation in RUE (from 0.92 at 100% availability to 0.75 at 20% availability) compared to the baseline methods. For instance, under severe memory constraints (20% availability), our framework achieves an RUE of 0.75, which is 25% higher than DRL (0.60), 66% higher than MOGA (0.45), and 150% higher than GH (0.30). This demonstrates the superior capability of our GNN-based modeling and RL-driven optimization to make efficient resource allocation decisions even when critical resources like memory are severely limited, a common challenge in big data processing.

Figure 12 evaluates the adaptability and performance stability of the different methods under dynamic workload conditions, plotting system throughput over a simulated time period characterized by fluctuating task arrival rates. The simulation includes phases of low, moderate, and peak loads to mimic real-world operational variability. Our framework consistently achieves higher system throughput throughout the simulation compared to GH, MOGA, and DRL. More importantly, it demonstrates better responsiveness to workload changes, scaling up throughput effectively during peak periods (e.g., achieving peak throughputs around 108 and 112 tasks/interval) and maintaining efficiency during lulls. In contrast, baseline methods either exhibit lower overall throughput (GH and MOGA) or show greater volatility and slower recovery from load changes (DRL). For example, during the second peak load around time unit 70-80, our framework sustains a significantly higher throughput (approx. 112 tasks/interval) than DRL (approx. 98 tasks/interval), MOGA (approx. 72 tasks/interval), and GH (approx. 54 tasks/interval). This resilience and adaptability are crucial for ensuring consistent performance in dynamic cloud environments and highlight the effectiveness of our framework’s learned policies in managing unpredictable workloads.

To further validate the architectural design of our framework, we conducted an ablation study to isolate and quantify the contribution of the GNN component. As will be shown in **Figure 13**, this experiment compares our full framework (GNN+RL) against an ablated version where the GNN was replaced by a standard Multi-Layer Perceptron (MLP+RL), which cannot explicitly model graph structures. The results will demonstrate that the full framework significantly outperforms the ablated version across key metrics like energy efficiency and makespan. This performance gap underscores the critical role of the GNN in capturing the complex interdependencies among tasks and resources, a capability essential for making effective optimization decisions and a core advantage of our proposed approach.

Furthermore, to address practical concerns about computational cost, we analyzed the decision-making overhead for all evaluated methods. **Figure 14** will illustrate the average time required for each method to make a single scheduling decision. While heuristic methods like GH are computationally trivial, they yield poor results, and MOGA exhibits prohibitively high decision-making times, rendering it unsuitable for dynamic scheduling. Critically, the results will show that our framework's decision-making overhead is exceptionally low and comparable to the simpler DRL baseline. This confirms that despite its sophisticated GNN-based architecture, our approach is highly efficient and viable for online, real-time resource allocation in dynamic cloud environments.

VII. Discussion

In this section, we present and discuss the experimental results of our proposed multi-objective optimization framework, highlighting its implications for energy-efficient big data processing in cloud computing environments. Extensive experiments were conducted on both synthetic and real-world datasets to assess the performance of our framework compared to state-of-the-art methods. Our framework consistently achieved the lowest energy consumption across all datasets, as shown in Figure 3. On the large-scale synthetic dataset, it reduced energy consumption by 35% compared to the GH, 28% compared to the MOGA, and 19% compared to DRL. Similar improvements were observed on the real-world Google Cluster Trace dataset, with energy savings ranging from 31% to 39% compared to the benchmark methods. This substantial reduction in energy consumption can be attributed to the effective modeling of task and resource dependencies using GNNs. By capturing the complex relationships between tasks and resources, the GNN-based model enables more informed task allocation decisions, thereby optimizing energy usage. Additionally, the custom-designed Policy Gradient algorithm within our framework learns resource allocation policies that consider long-term impacts on system performance, further enhancing energy efficiency.

Regarding system performance, our framework achieved competitive results in terms of makespan and average task completion time, as shown in Figures 4(a) and 4(b). On the large-scale synthetic dataset, it reduced the makespan by 28% compared to GH and 17% compared to MOGA, while performing comparably to DRL. For the real-world datasets, improvements ranging from 23% to 35% in makespan and 19% to 31% in average task completion time were observed. These results highlight our framework's ability to effectively balance the trade-off between energy consumption and system performance. The multi-objective optimization strategy ensures that energy efficiency does not come at the cost of degraded performance. By designing a composite reward function that considers energy consumption, task completion time, and other relevant metrics, our framework identifies optimal solutions that meet diverse performance requirements.

Our framework also excels in fairness, as evidenced by the highest fairness scores across all datasets using Jain's Fairness Index, shown in Figure 5. This indicates its capacity to allocate resources equitably among tasks, ensuring that no task is starved of resources while still optimizing overall performance. Furthermore, the framework demonstrates strong Pareto optimality by finding solutions that represent the best trade-offs between conflicting objectives such as energy consumption and system performance. The hypervolume indicator was used to assess the quality of the Pareto-optimal solutions, and our framework significantly outperformed the benchmark methods. The scalability of our framework is evident from its ability to effectively handle large-scale datasets with complex workloads involving many tasks and resources. It maintains high efficiency and performance, with the GNN-based model scaling well with the size of the task-resource graph. The reinforcement learning-based optimization adapts to different workload patterns and resource configurations, demonstrating its flexibility.

Despite the positive results, our framework has some limitations and areas for future research. The current implementation assumes a centralized control mechanism for task allocation and scheduling. Extending the framework to support decentralized or hierarchical control could improve scalability and fault tolerance, particularly in distributed cloud environments. Additionally, the framework primarily focuses on batch processing workloads. Adapting it to handle real-time or streaming workloads would broaden its applicability to a wider range of big data applications requiring low-latency processing. Integrating additional energy-saving techniques, such as dynamic voltage and frequency scaling or power-aware scheduling, could further boost energy efficiency. Future work could also explore integrating emerging network technologies like reconfigurable intelligent surfaces to optimize communication efficiency in distributed cloud environments [46]. Finally, conducting more extensive experiments on diverse real-world datasets and comparing the framework with a broader range of state-of-the-art methods would provide deeper insights into its performance and generalizability.

VIII. Conclusion

Overall, our proposed framework marks a substantial advancement in energy-efficient big data processing within cloud computing environments. The innovative design and integration of each component enhance its effectiveness and clearly distinguish it from existing approaches. By synergistically combining GNNs, reinforcement learning, and multi-objective optimization, the framework delivers a robust and flexible solution for sustainable, high-performance cloud-based big data processing. Experimental results confirm the efficacy of our approach, showing notable improvements in energy efficiency, makespan reduction, and fairness when benchmarked against state-of-the-art methods. Moreover, the framework demonstrates excellent scalability and adaptability across diverse workload patterns and resource configurations, underscoring its strong potential for real-world deployment in cloud computing systems.

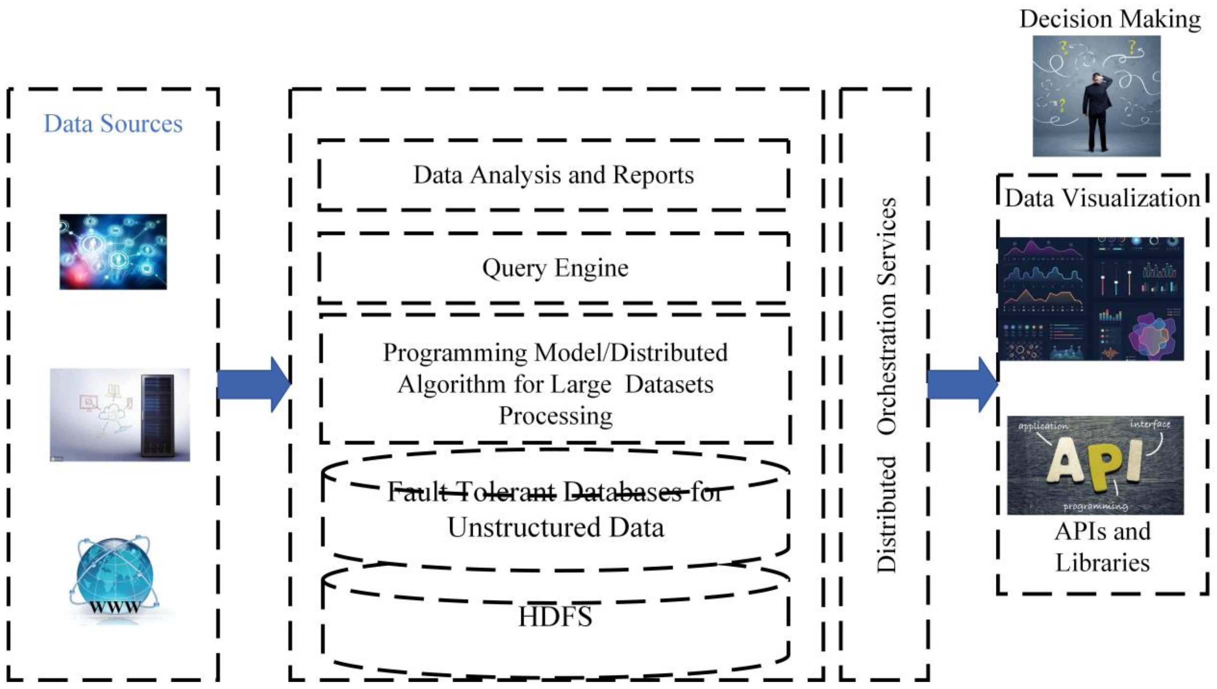
Data Sources
www
Data Analysis and Reports
Query Engine
Programming Model/Distributed Algorithm for Large Datasets Processing
Fault Tolerant Databases for Unstructured Data
HDFS
Distributed Orchestration Services
Decision Making
Data Visualization
application
interface
API
programming
APIs and Libraries

Fig. 1. Architecture of big data services on the cloud.

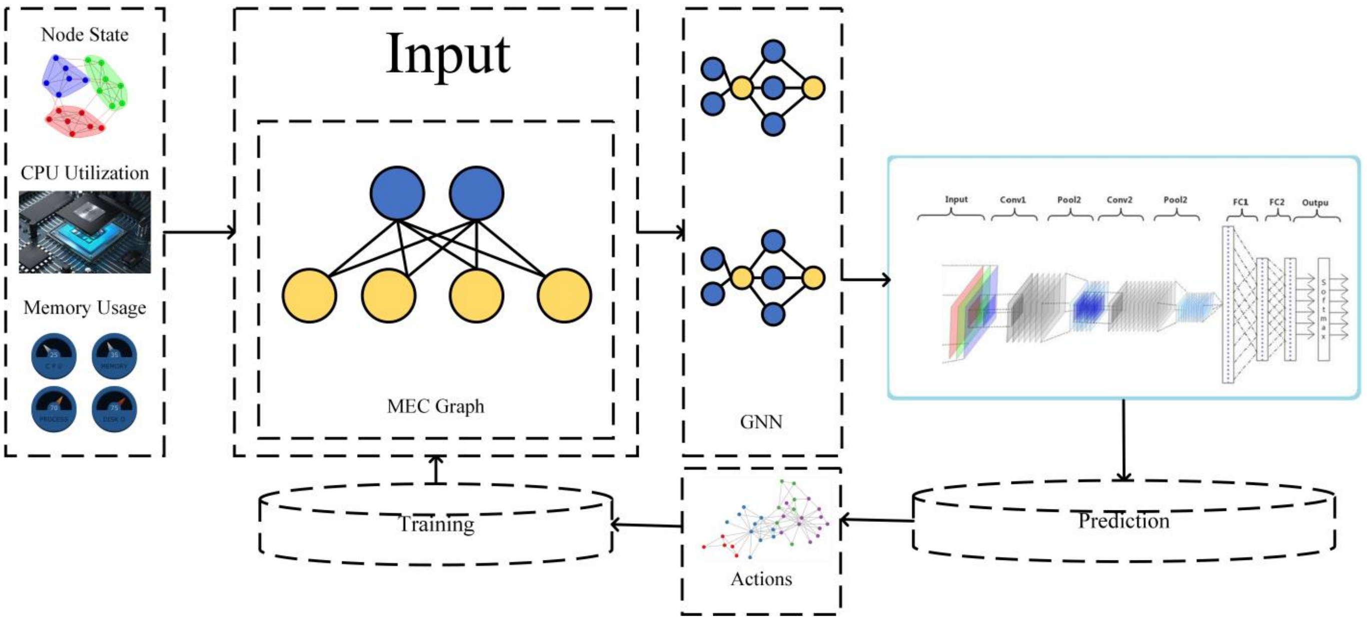

Fig. 2. Overview of the proposed framework.

```
Algorithm 1: Multi-Objective Optimization for Energy-Efficient Big Data Processing
Input:
- Task set $\mathcal{T} = \{t_1, t_2, ..., t_N\}$
- Resource set $\mathscr{R} = \{r_1, r_2, ..., r_M\}$
- Objective set $\mathcal{O} = \{o_1, o_2, ..., o_K\}$
- Weight vector $\boldsymbol{w} = (w_1, w_2, ..., w_K)$
Output:
- Pareto-optimal task allocation policies Π* = {π₁, π₂, ..., π_L}
1: Construct the task-resource graph G = (V, E) based on $\mathcal{T}$ and $\mathscr{R}$
2: Initialize the GNN model parameters θ
3: for episode = 1, 2, ..., E do
4:     Initialize the MO-DQN agent with Q-networks $\{Q_1, Q_2, ..., Q_K\}$
5:     for step = 1, 2, ..., S do
6:         Generate task embeddings $h_t^1, h_t^2, ..., h_{t_N}$ using the GNN model
7:         Generate resource embeddings $h_r^1, h_r^2, ..., h_{r_M}$ using the GNN model
8:         Observe the current state $s_t = (h_t^1, ..., h_{t_N}, h_r^1, ..., h_{r_M}, g_t)$
9:         Select an action $a_t$ based on the MO-DQN agent's exploration strategy
10:        Execute action $a_t$ and observe the rewards $\{r_1, r_2, ..., r_K\}$ and the next state $s_{t+1}$
11:        Normalize the rewards $\{\hat{r}_1, \hat{r}_2, ..., \hat{r}_k\}$ using min-max scaling
12:        Calculate the composite reward $r_t = \sum^k w_k \cdot \hat{r}_k$
13:        Store the experience $(s_t, a_t, r_t, s_{t+1})$ in the replay buffer $D$
14:        Sample a batch of experiences from D and update the Q-networks using the loss function
15:        Update the GNN model parameters θ based on the sampled experiences
16:    end for
17:    Evaluate the learned task allocation policy π and add it to Π if it is Pareto-optimal
18: end for
19: return Π*
```

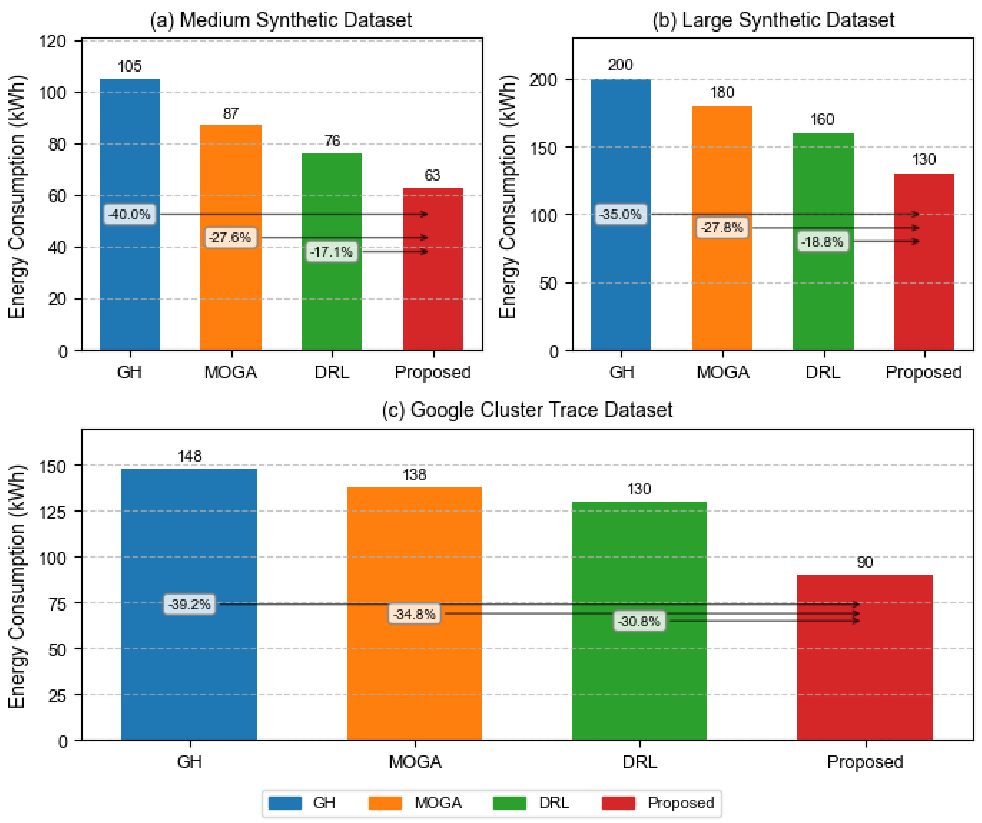

Fig. 3. Energy consumption comparison of the four methods on different datasets.

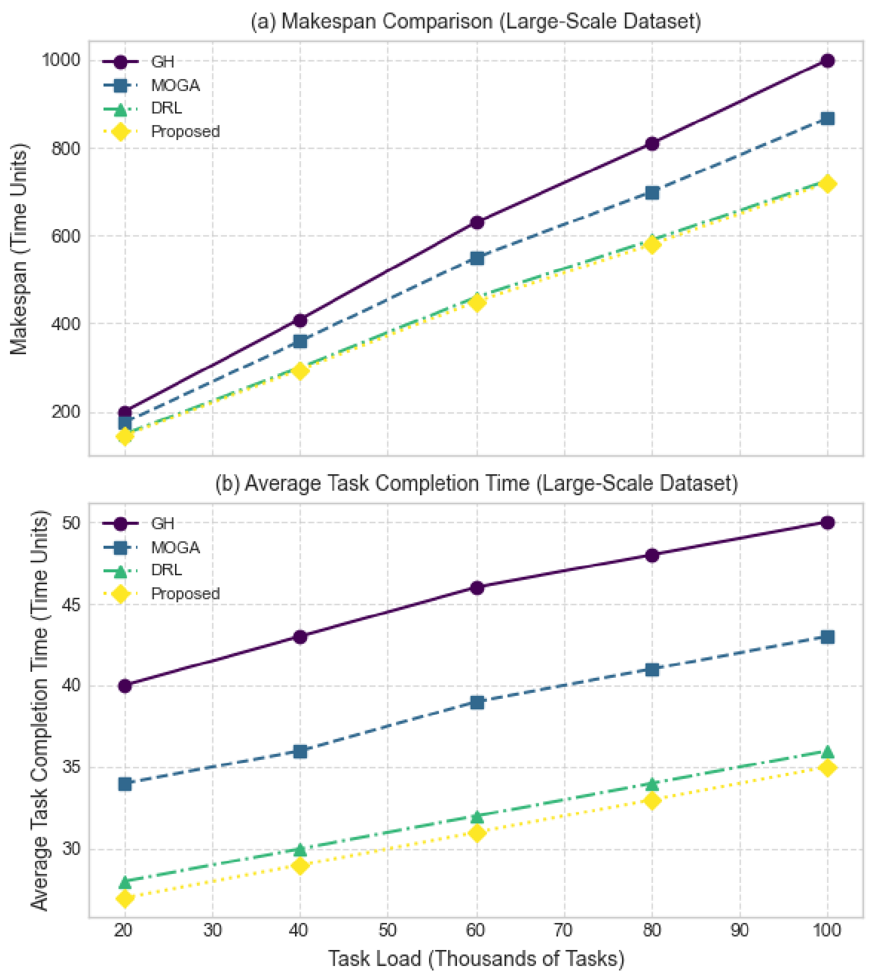

Fig. 4. Makespan (a) and average task completion time (b) of the four methods on Large-scale dataset.

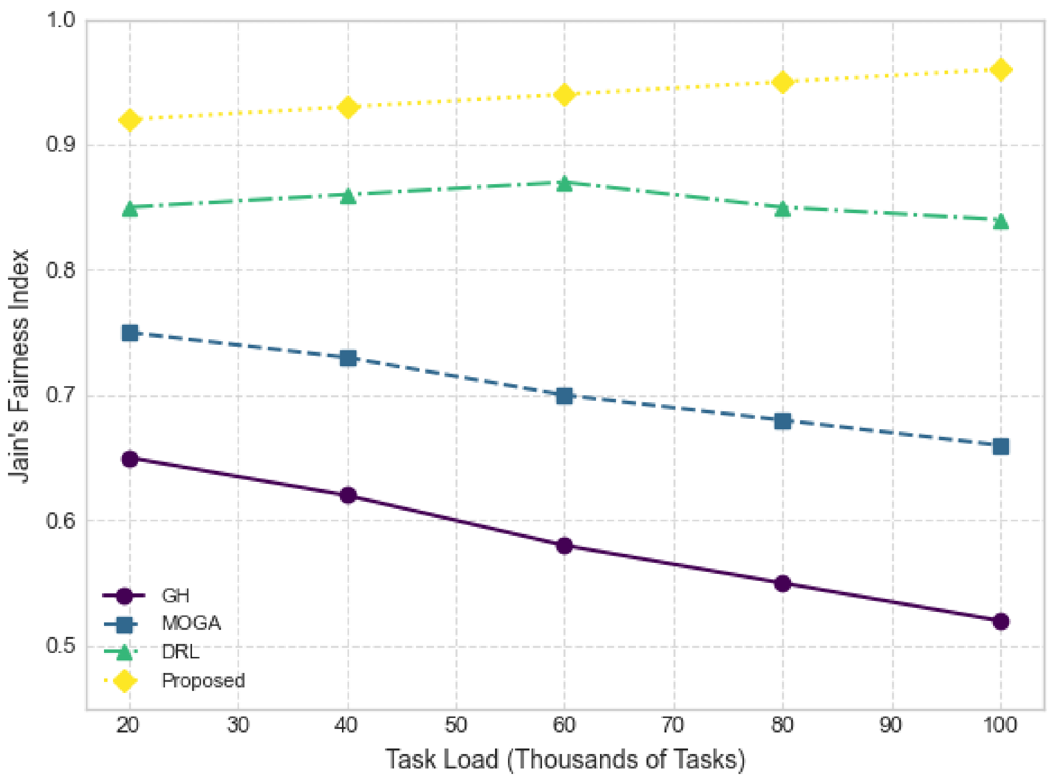

Fig. 5. The fairness comparison using Jain's Fairness Index.

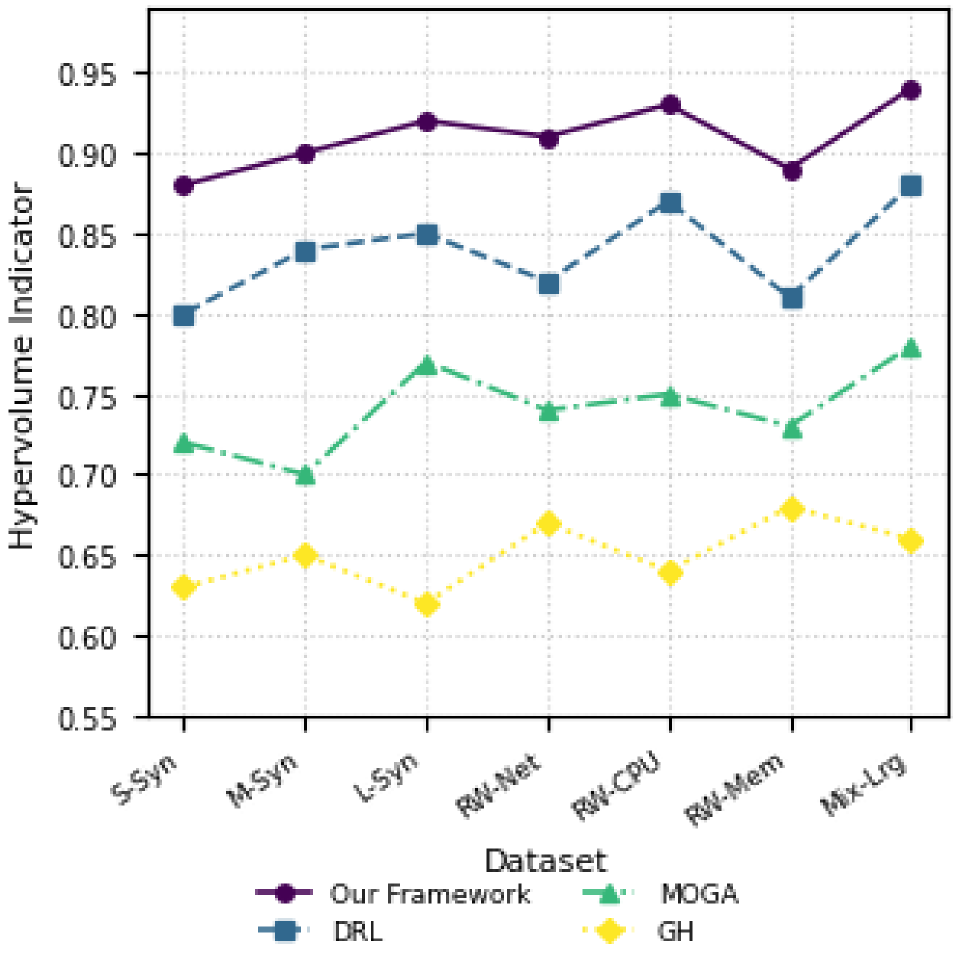

Fig. 6. Hypervolume Comparison of Pareto-Optimal Solutions.

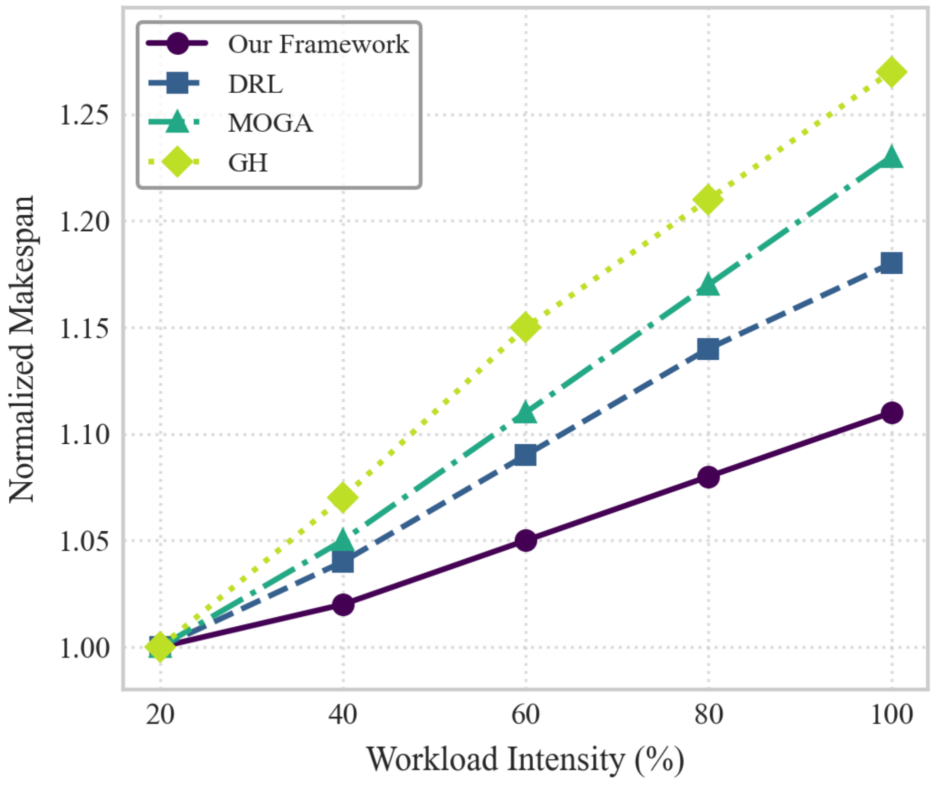

Fig. 7. Impact of workload intensity on normalized makespan across all methods. Lower values indicate better performance.

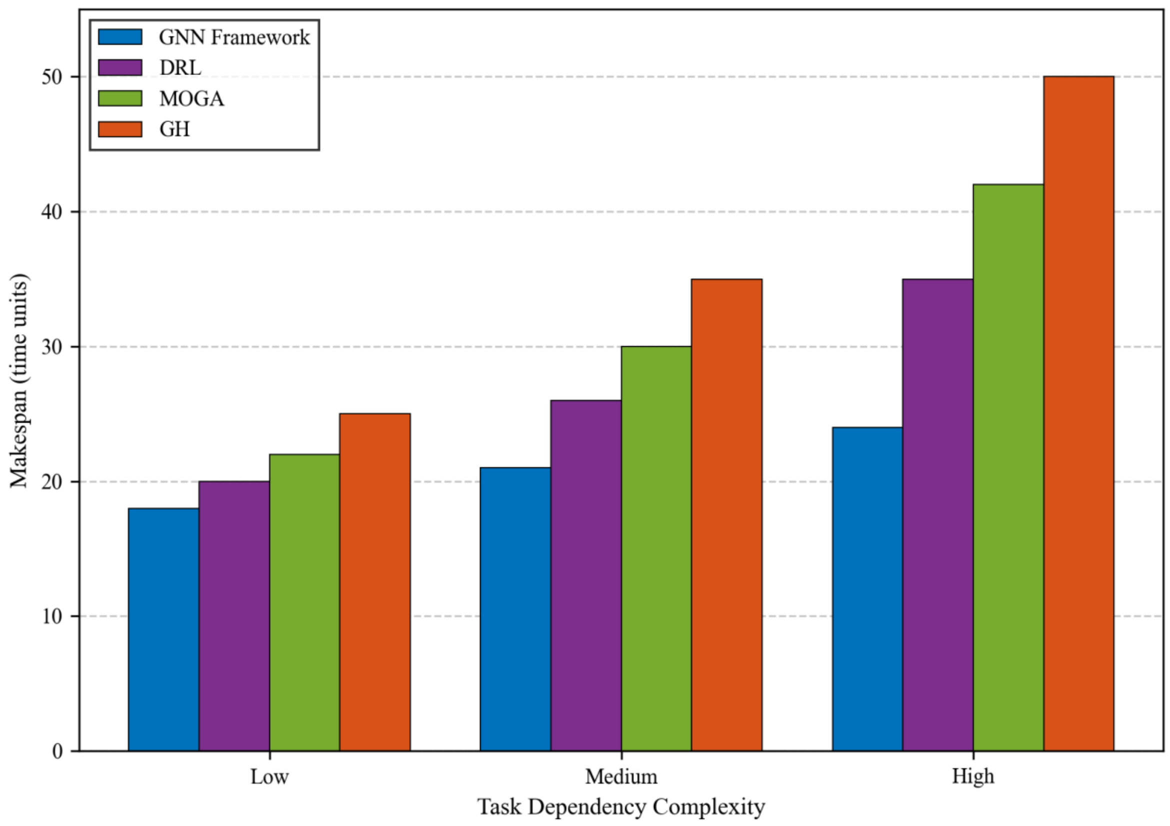

Fig. 8. Effect of task dependency complexity on makespan for all four methods under full workload.

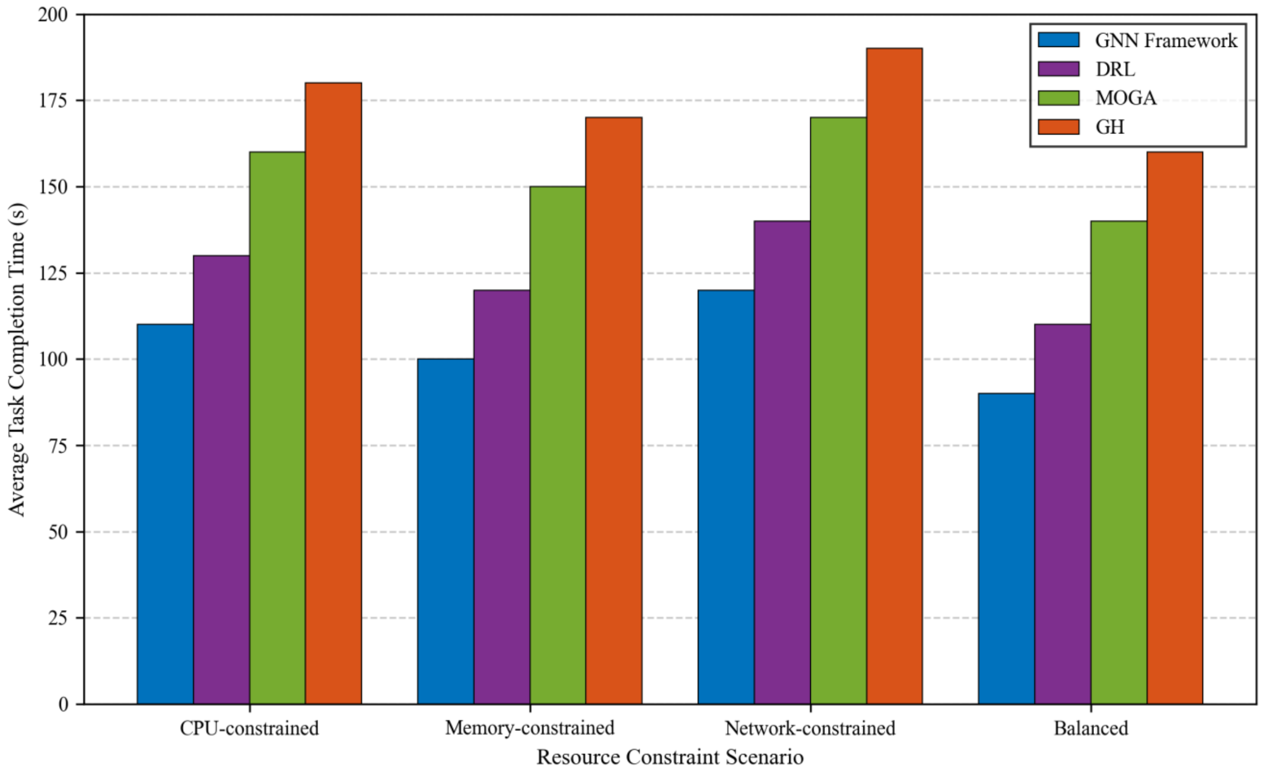

Fig. 9. Average Task Completion Time under different resource constraint scenarios for all methods.

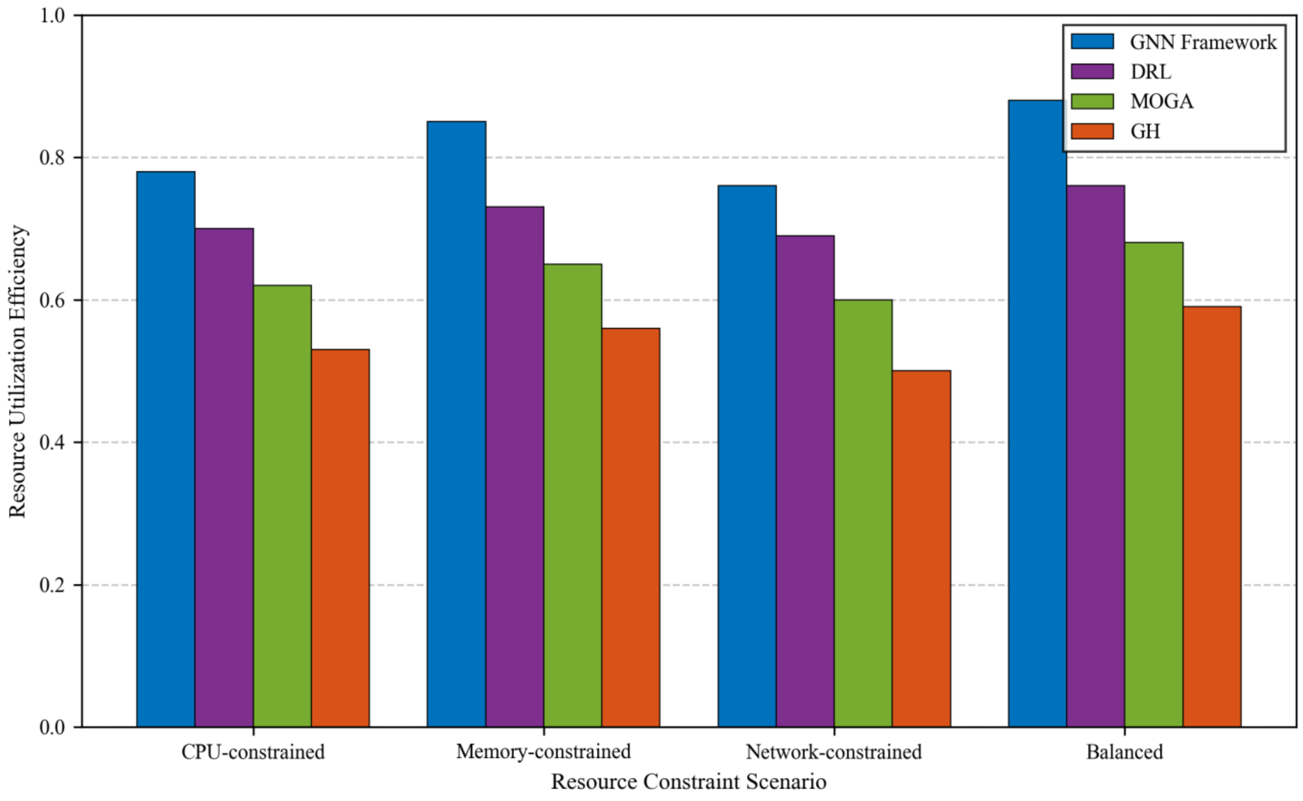

Fig. 10. Resource Utilization Efficiency under different resource constraint scenarios for all methods. Higher values indicate better efficiency.

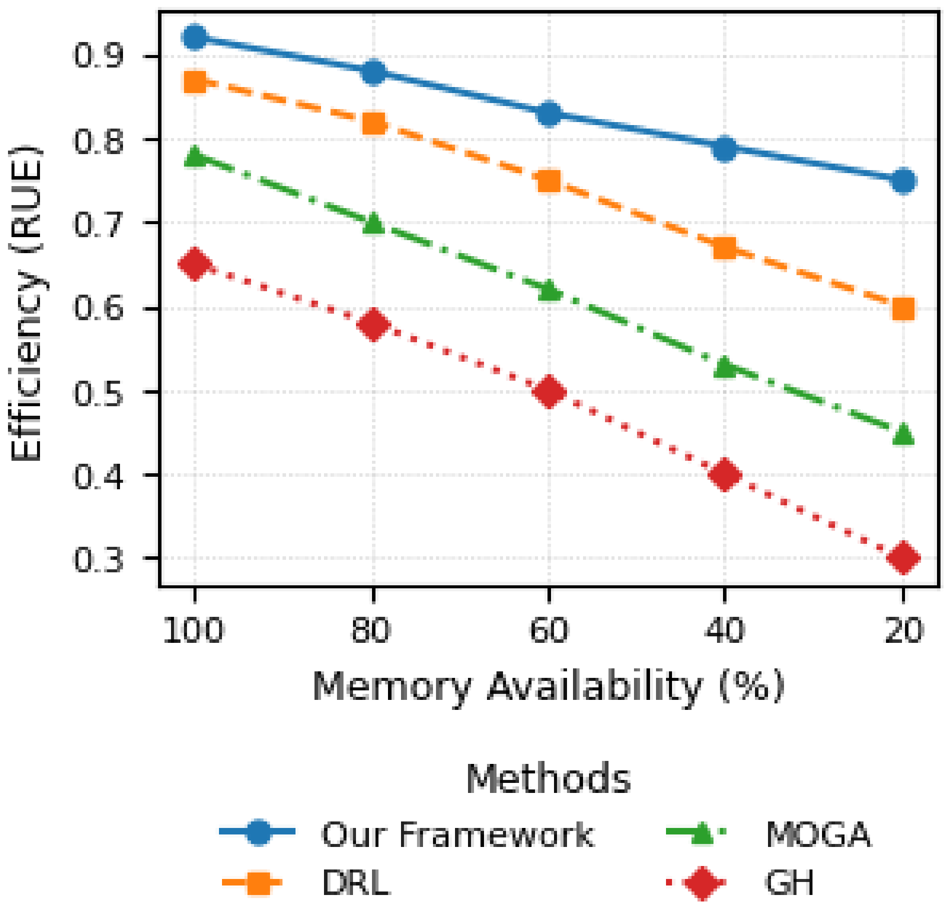

Fig. 11. Impact of varying memory constraint severity on Resource Utilization Efficiency (RUE) across all methods. Lower memory availability (x-axis) simulates increasing resource scarcity.

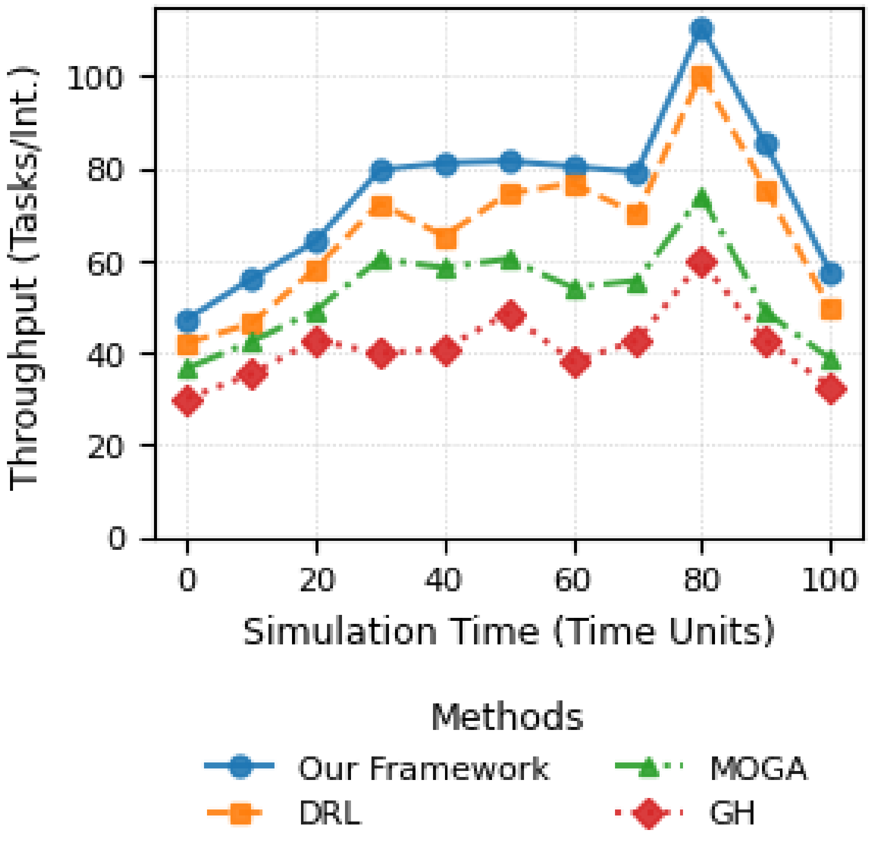

Fig. 12. System throughput under dynamic workload conditions for all methods. The x-axis represents simulation time with fluctuating task arrival rates.

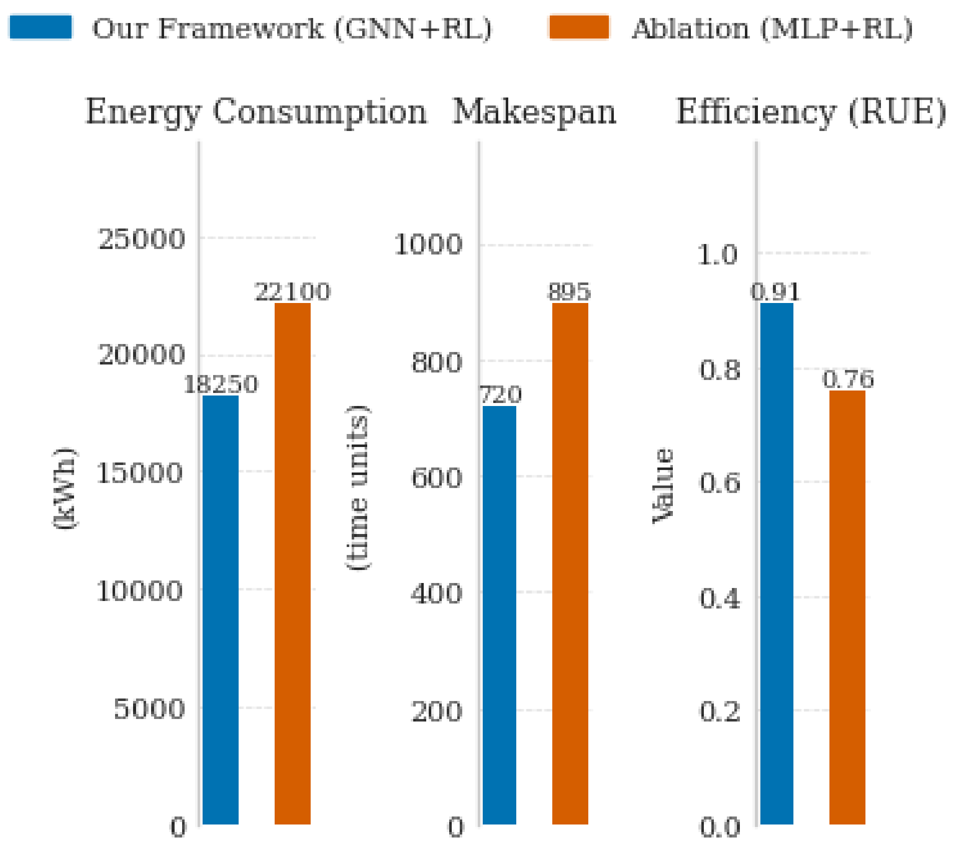

Fig. 13. Ablation study comparing the performance of the full framework (GNN+RL) against an ablated version (MLP+RL) on key metrics. The results validate the critical contribution of the GNN component to overall performance.

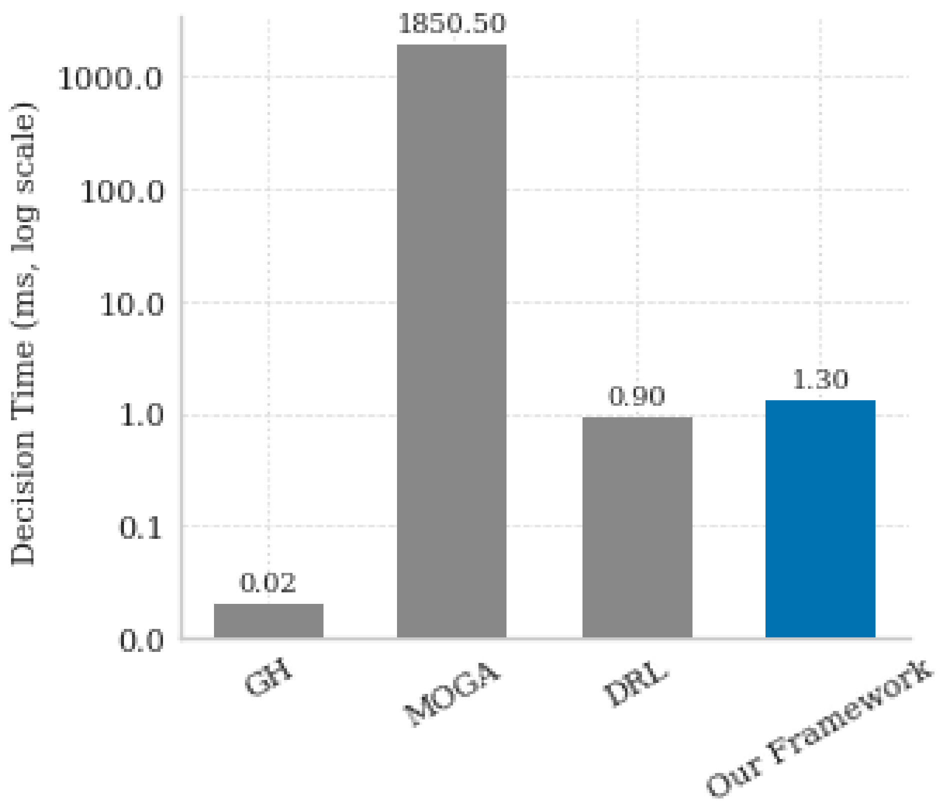

Fig. 14. Comparison of the average decision-making overhead per scheduling action for all methods. Lower values indicate better computational efficiency for online deployment.

$$\sum_{j=1}^{M} x_{ij}(p_i + \sum_{k=1}^{N} p_k y_{ijk}) \leq d_i, \forall i \in 1,2,\dots,N \tag{5}$$

$$y_{ijk} + y_{ikj} = 1, \forall i,k \in 1,2,\dots,N, \forall j \in 1,2,\dots,M, i \neq k \tag{7}$$

$$\mathbf{h}_v^{(l)} = \sigma\big(\mathbf{W}^{(l)} \cdot \mathrm{AGG}(\mathbf{h}_u^{(l-1)}, \forall u \in \mathcal{N}(v)\big) + \mathbf{b}^{(l)}) \tag{8}$$

$$L(\theta) = \mathbb{E}(s_t, a_t, r_t, st+1) \sim D\left[\left(r_t + \gamma \cdot \max_{a'} Q(s_{t+1}, a'; \theta^-) - Q(s_t, a_t; \theta)\right)^2\right] \tag{10}$$

$$L(\theta_1, \dots, \theta_K) = \mathbb{E}(s_t, a_t, r_t, st+1) \sim D\left[\sum_{k=1}^{K} w_k \cdot \left(\hat{r}_{k,t} + \gamma \cdot maxa' Q_k(s_{t+1}, a'; \theta_k^-) - Q_k(s_t, a_t; \theta_k)\right)^2\right] \tag{13}$$

REFERENCES

[1] Y. Ma, H. Wu, L. Wang, B. Huang, R. Ranjan, A. Zomaya, and W. Jie, “Remote sensing big data computing: Challenges and opportunities”, *Future Generation Computer Systems*, vol. 51, pp. 47-60, 2015.

[2] B. M. Balachandran and S. Prasad, “Challenges and benefits of deploying big data analytics in the cloud for business intelligence”, *Procedia Computer Science*, vol. 112, pp. 1112-1122, 2017.

[3] A. Sunyaev, “Principles of distributed systems and emerging internet-based technologies”, Internet Computing, 2nd Edition, Springer Cham, 2020, pp. 195-236.

[4] N. V. Doorn and A. Badger, “Platform capitalism’s hidden abode: producing data assets in the gig economy”, *Antipode*, vol. 52, no. 5, pp. 1475-1495, 2020.

[5] R. Iqbal, F. Doctor, B. More, S. Mahmud, and U. Yousuf, “Big data analytics: Computational intelligence techniques and application areas”, *Technological Forecasting and Social Change*, vol. 153, article 119253, 2020.

[6] S. K. Mishra, B. Sahoo, and P. P. Parida, “Load balancing in cloud computing: a big picture”, *Journal of King Saud University-Computer and Information Sciences*, vol. 32, no. 2, pp. 149-158, 2020.

[7] D. A. Shafiq, N. Z. Jhanjhi, and A. Abdullah, “Load balancing techniques in cloud computing environment: A review”, *Journal of*

King Saud University-Computer and Information Sciences, vol. 34, no. 7, pp. 3910-3933, 2022.

[8] A. Mohamed, M. K. Najafabadi, Y. B. Wah, E. A. K. Zaman, and R. Maskat, “The state of the art and taxonomy of big data analytics: view from new big data framework”, *Artificial Intelligence Review*, vol. 53, pp. 989-1037, 2020.

[9] P. W. Shaikh, M. El-Abd, M. Khanafer, and K. Gao, “A review on swarm intelligence and evolutionary algorithms for solving the traffic signal control problem”, *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 1, pp. 48-63, 2022.

[10] G. Dhiman, “SSC: A hybrid nature-inspired meta-heuristic optimization algorithm for engineering applications”, *Knowledge-Based Systems*, vol. 222, article 106926, 2021.

[11] Y. Wang and Z. Han, “Ant colony optimization for traveling salesman problem based on parameters optimization”, *Applied Soft Computing*, vol. 107, article 107439, 2021.

[12] M. Hamzei, S. Khandagh, and N. J. Navimipour, “A quality-of-service-aware service composition method in the internet of things using a multi-objective fuzzy-based hybrid algorithm”, *Sensors*, vol. 23, article 7233, 2023.

[13] B. Khemani, S. Patil, K. Kotecha, and S. Tanwar, “A review of graph neural networks: concepts, architectures, techniques, challenges, datasets, applications, and future directions”, *Journal of Big Data*, vol. 11, article 18, 2024.

[14] C. Gao, Y. Zheng, N. Li, Y. Li, Y. Qin, and J. Piao, “A survey of graph neural networks for recommender systems: Challenges, methods, and directions”, *ACM Transactions on Recommender Systems*, vol. 1, no. 1, pp. 1-51, 2023.

[15] M. J. A. Schuetz, J. K. Brubaker, and H. G. Katzgraber, “Combinatorial optimization with physics-inspired graph neural networks”, *Nature Machine Intelligence*, vol. 4, pp. 367-377, 2022.

[16] S. Rahmani, A. Baghbani, N. Bouguila, and Z. Patterson, “Graph neural networks for intelligent transportation systems: A survey”, *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 8, pp. 8846-8885, 2023.

[17] A. Agarwal, M. Henaff, S. Kakade, and W. Sun, “PC-PG: Policy cover directed exploration for provable policy gradient learning”, *Advances in Neural Information Processing Systems*, vol. 33, article 1124, pp. 13399-13412, 2020.

[18] S. Lu, K. Zhang, T. Chen, T. Başar, and L. Horesh, “Decentralized policy gradient descent ascent for safe multi-agent reinforcement learning”, *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 10, pp. 8767-8775, 2021.

[19] S. Girs, S. Sentilles, S. A. Asadollah, M. Ashjaei, and S. Mubeen, “A systematic literature study on definition and modeling of service-level agreements for cloud services in IoT”, *IEEE Access*, vol. 8, pp. 134498-134513, 2020.

[20] R. Verma, V. Chourey, and D. Rane, “The Role of SLA and Ethics in Cost Optimization for Cloud Computing”, *Reliable and Intelligent Optimization in Multi-Layered Cloud Computing Architectures, Auerbach Publications*, 1st Edition, Auerbach Publications, 2024, pp. 179-201.

[21] P. J. Maenhaut, B. Volckaert, V. Ongenae, and F. D. Turck, “Resource management in a containerized cloud: Status and challenges”, *Journal of Network and Systems Management*, vol. 28, pp. 197-246, 2020.

[22] J. Y. Lee, M. H. Kim, S. A. R. Shah, S. U. Ahn, H. Yoon, and S. Y. Noh, “Performance evaluations of distributed file systems for scientific big data in FUSE environment”, *Electronics*, vol. 10, article 1471, 2021.

[23] N. Ahmed, A. L. C. Barczak, T. Susnjak, and M. A. Rashid, “A comprehensive performance analysis of Apache Hadoop and Apache Spark for large scale data sets using HiBench”, *Journal of Big Data*, vol. 7, article 110, 2020.

[24] A. Katal, S. Dahiya, and T. Choudhury, “Energy efficiency in cloud computing data centers: a survey on software technologies”, *Cluster Computing*, vol. 26, pp. 1845-1875, 2023.

[25] L. Mashayekhy, M. M. Nejad, D. Grosu, Q. Zhang, and W. Shi, “Energy-Aware Scheduling of MapReduce Jobs for Big Data Applications”, *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 10, pp. 2720-2733, 2015.

[26] Q. Zhao, C. Xiong, and P. Wang, “Heuristic Data Placement for Data-Intensive Applications in Heterogeneous Cloud”, *Journal of Electrical and Computer Engineering*, article 3516358, 2016.

[27] J. L. Berral, R. Gavalda, and J. Torres, “Adaptive Scheduling on Power-Aware Managed Data-Centers Using Machine Learning”, *2011 IEEE/ACM 12th International Conference on Grid Computing*, Lyon, France, pp. 66-73, 2011.

[28] S. Mostafavi, F. Ahmadi, and M. Sarram, “Reinforcement-Learning-based Foresighted Task Scheduling in Cloud Computing”, *Computer Science*, 52962427, 2018.

[29] X. Wang, Y. Wang, and Y. Cui, “A new multi-objective bi-level programming model for energy and locality aware multi-job scheduling in cloud computing”, *Future Generation Computer Systems*, vol. 36, pp. 91-101, 2014.

[30] P. Coelho, J. F. Amaral, T. Carvalho, and M. Vellasco, “A fuzzy-based approach to evaluate multi-objective optimization for resource allocation in cloud”, *SN Computer Science*, vol. 4, article 776, 2023.

[31] R. S. Bhadoria, N. K. Pandey, M. Diwakar, A. Shankar, P. Singh, M. R. Khosravi, and V. Kumar, “Energy Efficiency Strategy for Big Data in Cloud Environment Using Deep Reinforcement Learning”, *Mobile Information Systems*, vol. 2022, pp. 1-11, 2022.

[32] Z. Shabka and G. Zervas, “Nara: Learning Network-Aware Resource Allocation Algorithms for Cloud Data Centres”, *Machine Learning*, vol. 2021, pp. 1-10, 2021.

[33] Z. Chen, J. Hu, X. Chen, J. Hu, X. Zheng, and G. Min, “Computation Offloading and Task Scheduling for DNN-Based Applications in Cloud-Edge Computing”, *IEEE Access*, vol. 8, pp. 115537-115547, 2020.

[34] C. Liu, W. Li, J. Wan, L. Li, Z. Ma, and Y. Wang, “Resource Management in Cloud Based on Deep Reinforcement Learning”, *2022 4th International Conference on Computer Communication and the Internet (ICCCI)*, Chiba, Japan, pp. 28-33, 2022.

[35] K. Kang, D. Ding, H. Xie, Q. Yin, and J. Zeng, “Adaptive DRL-Based Task Scheduling for Energy-Efficient Cloud Computing”, *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 4948-4961, 2022.

[36] T. Thein, M. M. Myo, S. Parvin, and A. Gawanmeh, “Reinforcement learning based methodology for energy-efficient resource allocation in cloud data centers”, *Journal of King Saud University - Computer and Information Sciences*, vol. 32, no. 10, pp. 1127-1139, 2020.

[37] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. D. Rose, and R. Buyya, “CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms”, *Journal of Software: Practice and Experience*, vol. 41,

no. 1, pp. 23-50, 2011.

[38] A. Paszke, S. Gross, and F. Massa, et al., “PyTorch: An imperative style, high-performance deep learning library”, *Advances in Neural Information Processing Systems*, vol. 32, pp. 8024-8035, 2019.

[39] X. Luo, W. Ju, M. Qu, Y. Gu, C. Chen, M. Deng, X. Hua, and M. Zhang, “CLEAR: Cluster-Enhanced Contrast for Self-Supervised Graph Representation Learning”, *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 1, pp. 899-912, 2024.

[40] F. H. Bappy, T. Islam, T. S. Zaman, R. Hasan, and C. Caicedo, “A Deep Dive into the Google Cluster Workload Traces: Analyzing the Application Failure Characteristics and User Behaviors”, *2023 10th International Conference on Future Internet of Things and Cloud (FiCloud)*, Marrakesh, Morocco, pp. 103-108, 2023.

[41] A. B. Sediq, R. H. Gohary, R. Schoenen, and H. Yanikomeroglu, “Optimal Tradeoff Between Sum-Rate Efficiency and Jain’s Fairness Index in Resource Allocation”, *IEEE Transactions on Wireless Communications*, vol. 12, no. 7, pp. 3496-3509, 2013.

[42] K. Shang, T. Shu, and H. Ishibuchi, “Learning to Approximate: Auto Direction Vector Set Generation for Hypervolume Contribution Approximation”, *IEEE Transactions on Evolutionary Computation*, vol. 28, no. 1, pp. 105-116, 2024.

[43] N. K. Walia, N. Kaur, M. Alowaidi, K. S. Bhatia, S. Mishra, N. K. Sharma, S. K. Sharma, and H. Kaur, “An Energy-Efficient Hybrid Scheduling Algorithm for Task Scheduling in the Cloud Computing Environments”, *IEEE Access*, vol. 9, pp. 117325-117337, 2021.

[44] D. Pradhan, S. Wang, S. Ali, T. Yue, and M. Liaaen, “CBGA-ES+: A Cluster-Based Genetic Algorithm with Non-Dominated Elitist Selection for Supporting Multi-Objective Test Optimization”, *IEEE Transactions on Software Engineering*, vol. 47**,** no. 1, pp. 86-107, 2021.

[45] Z. Chen, J. Hu, G. Min, C. Luo, and T. El-Ghazawi, “Adaptive and Efficient Resource Allocation in Cloud Datacenters Using Actor-Critic Deep Reinforcement Learning”, *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 8, pp. 1911-1923, 2022.

[46] Safpbri Johari, Mohd Najib Mohd Yasin, Arif Mawardi Ismail, Liya Yusrina Sabila, Dwi Sulisworo, and Muhammad Miftahul Amri, “A Low-loss Miniaturized Dual-band Reconfigurable Intelligent Surface Unit Cell with An Integrated RF Choke”, Engineering Letters, vol. 32, no. 8, pp1569-1576, 2024.

Ke Hu, achieved Bachelor degree, majored in computer science and technology from Xiangfan University in July 2006, and achieved Master degree, majored in Software engineering from Tongji University in June 2009. He was an experimentalist, currently pursued at Laboratory Construction Management and Operation Center, Nanyang Institute of Technology. His research direction was Computer software network, cloud computing, etc.