Computing (2026) 108:67
https://doi.org/10.1007/s00607-026-01655-8

REGULAR ARTICLE

AI-driven resource allocation in cloud computing: a systematic review revealing critical sustainability and evaluation gaps

Shahzeb Alam[1] · Muhammad Atif Ramzan[2] · Muhammad Zubair[3] · Ubaid Ullah[4] · Ziaullah[5] · Rab Nawaz[6] · Rahmat Ullah[6]

Received: 8 September 2025 / Accepted: 18 March 2026

Abstract

This systematic literature review analyzes AI-driven resource allocation in cloud computing through comprehensive analysis of 63 high-quality studies selected via PRISMA 2020 methodology from an initial collection of 485 papers. Our taxonomic framework categorizes approaches across four dimensions: algorithmic methods, deployment environments, optimization objectives, and evaluation methods. Quantitative analysis demonstrates substantial AI superiority over traditional approaches: ≈45% average latency reduction (range 11−77.7%) from 10 studies with quantifiable latency data, 32% cost savings (range 10–48%) from 6 studies with quantifiable cost data, and 35% energy efficiency improvements (range 3.68–71%) from 16 studies with energy measurements. Reinforcement learning dominates the field (40% of studies) with particular effectiveness in dynamic environments, while hybrid approaches demonstrate superior multi-objective optimization. Critical research gaps include minimal carbon-aware scheduling integration (only 4 studies, 6.3% of corpus), over-reliance on simulation environments (70% of evaluations), and absence of standardized evaluation frameworks. The limited availability of quantifiable performance data across studies reveals a significant methodological gap in current research evaluation practices. We identify five high-priority research directions and provide actionable recommendations for advancing production-ready AI-driven cloud resource management systems that balance performance, sustainability, and practical deployment requirements.

Keywords Cloud computing · Resource allocation · Artificial intelligence · Machine learning · Sustainability · Container orchestration · Serverless computing · Multi-agent systems

Mathematics Subject Classification 68M20 · 68T05

Extended author information available on the last page of the article

Published online: 16 April 2026

1 Introduction

Cloud computing resource allocation faces unprecedented complexity as workloads become increasingly dynamic, infrastructure grows heterogeneous, and service-level objectives (SLOs) become more stringent [1–3]. Traditional static provisioning and rule-based autoscaling mechanisms struggle with modern cloud-native environments, particularly in container orchestration platforms like Kubernetes and serverless computing frameworks [4, 5]. These conventional approaches exhibit fundamental limitations: reactive rather than proactive decision-making, inability to predict workload patterns, and challenges in optimizing across multiple conflicting objectives simultaneously. This has driven extensive research into artificial intelligence (AI) and machine learning (ML) approaches for autonomous, adaptive resource management.

Cloud resource management has evolved through three distinct phases: traditional static provisioning (pre-2015), machine learning integration (2015–2020), and current AI-driven optimization (2020–2025). This evolution reflects the field's progression from reactive threshold-based systems to sophisticated AI frameworks leveraging deep learning, reinforcement learning, and hybrid approaches [6–11]. Furthermore, container orchestration, dominated by Kubernetes, and serverless computing paradigms such as Function-as-a-Service (FaaS) present unique resource allocation challenges. Kubernetes requires sophisticated scheduling across heterogeneous clusters, while serverless platforms must address cold start latency and function placement optimization [5, 7, 12, 13].

Recent studies report 30–71% latency reductions, 10–44% cost savings, and 22–71% energy efficiency improvements over traditional approaches [14–16], particularly in dynamic environments. Despite these advances, significant gaps persist. Our systematic review identifies both the potential benefits and critical limitations of current approaches in carbon-aware scheduling (only 6.3% of studies), over-reliance on simulation environments (70% of evaluations), and absence of standardized evaluation frameworks.

This systematic review advances the field through three significant contributions:

1. **Comprehensive Performance Analysis:** Systematic synthesis of quantifiable performance data revealing AI advantages over traditional approaches and evaluation methodology gaps.
2. **Critical Gap Identification:** Evidence-based identification of sustainability research limitations (only 6.3% address carbon-awareness) and evaluation methodology weaknesses (70% simulation-only)
3. **Research Roadmap:** Prioritized framework of five research directions with specific recommendations for practitioners and researchers

The remainder of this paper presents our systematic methodology in Sect. 3, comprehensive taxonomy in Sect. 4, quantitative results in Sect. 5, critical analysis in Sect. 6, and future research directions in Sect. 7.

2 Related work and background

Cloud resource allocation encompasses five interconnected processes: provisioning, scheduling, autoscaling, load balancing, and consolidation. Traditional approaches rely on static thresholds and reactive policies that fail to adapt to dynamic workloads, leading to either over-provisioning waste or under-provisioning degradation [4, 14].

2.1 Resource management decision processes

Resource provisioning determines the initial allocation of computational resources based on anticipated demand patterns [4, 6]. Traditional provisioning approaches rely on static capacity planning and historical usage patterns, leading to either resource waste through over-provisioning or performance degradation through under-provisioning in dynamic cloud environments [1, 14]. This challenge intensifies in container orchestration platforms where workload characteristics exhibit high temporal variability and unpredictable scaling patterns. Rule-based methods use predefined thresholds, providing simplicity but exhibiting reactive behavior [15, 17].

Task scheduling assigns computational tasks to available resources while optimizing multiple competing objectives including latency minimization, cost efficiency, and energy consumption [2, 18]. Conventional deterministic scheduling (first-come-first-served, round-robin, priority-based) fails to adapt to changing conditions [5, 12], particularly in containerized environments with dynamic resource requirements. These limitations become particularly pronounced in containerized environments where task granularity varies significantly and resource requirements change dynamically. Heuristic algorithms (first-fit, best-fit, greedy) provide reasonable solutions but lack learning capabilities for performance improvement.

Autoscaling mechanisms automatically adjust resource capacity based on observed demand patterns. However, threshold-based approaches exhibit three critical limitations: reactive decision-making that responds to load changes only after performance degradation occurs, single-metric optimization that ignores cost-performance trade-offs, and inability to anticipate future demand based on learned patterns [15, 19]. These constraints severely impact performance in serverless computing environments where cold start latency and function placement optimization create additional complexity layers [16, 20].

Load balancing distributes incoming requests across available resources to optimize system utilization and response times. Traditional algorithms including round-robin, least-connections, and weighted fair queuing lack awareness of heterogeneous resource capabilities, dynamic workload characteristics, and application-specific performance requirements [3, 21]. This limitation results in suboptimal resource utilization and performance degradation in modern multi-tenant cloud environments.

Resource consolidation optimizes infrastructure utilization by co-locating compatible workloads while maintaining performance isolation. This process requires sophisticated understanding of resource interference patterns, application dependencies, and performance correlation effects that exceed the analytical capabilities of rule-based approaches [14]. Traditional consolidation relies on simplistic metrics, while mathematical optimization (linear/integer programming) provides theoretical

optimality but faces scalability challenges [14]. Metaheuristics require parameter tuning and lack adaptability.

2.2 Container orchestration challenges

Kubernetes orchestration comprises four components: Scheduler (pod placement), Horizontal Pod Autoscaler (replica scaling), Vertical Pod Autoscaler (resource adjustment), and Cluster Autoscaler (node management) [4, 22]. Despite widespread adoption, this architecture exhibits fundamental limitations in handling complex workload patterns, multi-objective optimization scenarios, and proactive scaling decisions. These manifest as: reactive scaling causing performance degradation during spikes, single-metric optimization ignoring trade-offs, and static policies failing to adapt to workload evolution and changing application requirements over time, reducing operational efficiency in dynamic environments [23, 24].

2.3 Serverless computing resource challenges

Serverless platforms including AWS Lambda, Azure Functions, Google Cloud Functions, and Knative abstract infrastructure management through event-driven execution models and automatic scaling capabilities. However, they face four distinct resource allocation challenges that traditional approaches cannot adequately address [5, 12]. Moreover, cold start latency represents a critical performance bottleneck, with initialization delays ranging from 100–1500ms depending on runtime environment and function complexity [13]. Resource fragmentation emerges from fine-grained function execution patterns that create inefficient resource utilization across distributed infrastructure. Placement optimization across geographically distributed edge locations requires an understanding of network topology, data locality constraints, and latency requirements that exceed simple load balancing capabilities. The problems of anticipating massive workloads arise from unpredictable invocation patterns and variable execution characteristics inherent in serverless applications [7].

2.4 AI-driven solution advantages

AI methods address these limitations: reinforcement learning for sequential decision-making [1], deep learning provides pattern recognition for workload prediction [3, 21], and hybrid approaches for constraint satisfaction [12, 14], enabling proactive decision-making, and support multi-objective optimization without manual tuning. Prior surveys examine specific aspects such as machine learning [1] and deep RL methods [6] or container orchestration [4], but lack cross-cutting analysis. This review distinguishes itself through three contributions: (1) integrated taxonomy spanning methods, environments, objectives, and evaluation approaches; (2) quantified sustainability gap revealing only 6.3% carbon-aware studies with 74.6% ignoring energy metrics; (3) systematic evaluation critique showing 70% simulation reliance with only 6% production deployments.

3 Research methodology

This systematic literature review follows PRISMA 2020 guidelines [25] and systematic review methodology by [26] to ensure transparency, reproducibility, and methodological rigor.

3.1 Research questions

This systematic review addresses six research questions designed to evaluate AI-driven resource allocation:

- **RQ1 (Performance Comparison):** hat quantitative improvements do AI/ML methods achieve over traditional strategies across latency, cost, and energy?
- **RQ2 (Technical Approaches):**Which AI techniques demonstrate effectiveness in container orchestration and serverless autoscaling?
- **RQ3 (Evaluation Metrics):** What metrics and methodologies provide reliable evidence of AI-driven allocation effectiveness?
- **RQ4 (Limitations Analysis):** What limitations constrain practical deployment of current AI-driven approaches?
- **RQ5 (Sustainability Gap):** What gaps exist in carbon-aware scheduling and energy efficiency optimization?
- **RQ6 (Future Directions):** What research priorities are required for production-ready systems?

3.2 Search strategy and database selection

We searched five academic databases: IEEE Xplore (computer systems conferences and journals), ACM Digital Library (distributed systems venues), SpringerLink (AI publications), ScienceDirect (interdisciplinary research), and Google Scholar (preprints and additional sources). The search strategy employed a three-tier Boolean query combining AI methods, cloud platforms, and optimization objectives. Searches occurred March-May 2024 with systematic documentation yielding 485 initial papers. Our search strategy targeted three key domains: Firstly, AI methods (reinforcement learning, deep learning, hybrid approaches). Secondly, cloud platforms (kubernetes, serverless, edge computing), and thirdly, performance metrics (latency, cost, energy efficiency). The search employed a three-tier Boolean query structure:

Tier 1—AI Methods: “AI-driven resource allocation," “machine learning resource management," “deep reinforcement learning," “hybrid AI methods," “intelligent autoscaling"

Tier 2—Cloud Platforms: “cloud computing," “container orchestration," “Kubernetes," “serverless computing," “function-as-a-service"

Tier 3—Performance Metrics: “latency optimization," “cost efficiency," “energy efficiency," “carbon-aware scheduling," “resource utilization"

The final query structure was: (M_1 OR M_2 OR ... OR M_5) AND (P_1 OR P_2 OR ... OR P_5) AND (O_1 OR O_2 OR ... OR O_5), constrained to English-language publications from 2015–2025. Database-specific adaptations included IEEE Xplore

advanced field restrictions with publication date constraints and subject area filtering. ACM digital library searches employed computing classification system (CCS) taxonomy filtering for relevant categories including “Computing methodologies → Machine learning" and “Computer systems organization → Cloud computing."

3.3 Study selection process

Following duplicate removal (329 eliminated), 156 unique papers underwent title and abstract screening using predefined inclusion criteria. Two reviewers independently assessed study selection for quality assurance, achieving Cohen $K = 0.82$ indicating substantial agreement. Full-text evaluation of 89 papers applied rigorous exclusion criteria, with citation chaining adding 1 study, producing our final corpus of 63 studies as detail is giving in Fig. 1.

3.4 Study selection and data extraction

Inclusion criteria: peer-reviewed papers (2015–2025) on AI/ML for cloud resource allocation in container/serverless environments with empirical evaluation. **Exclusion criteria:** non-peer-reviewed work (except high-quality arXiv), network-only studies, surveys without contributions, and theoretical models without validation. Quality assessment covered methodological soundness, reproducibility, evaluation comprehensiveness, and practical relevance, plus AI-specific criteria (statistical validation, baseline quality, evaluation scale). Cohen’s K = 0.82 between two independent reviewers.

This systematic process identified 32 of 63 studies (51%) providing quantifiable performance data meeting rigorous analysis standards, with specific measurements available for latency (n = 10), cost (n = 6), and energy (n = 16). Current limitation is that the review protocol was not pre-registered in PROSPERO, though methodology strictly follows PRISMA 2020 guidelines with documented procedures and dual-reviewer assessment (Cohen’s K = 0.82). The remaining studies contributed

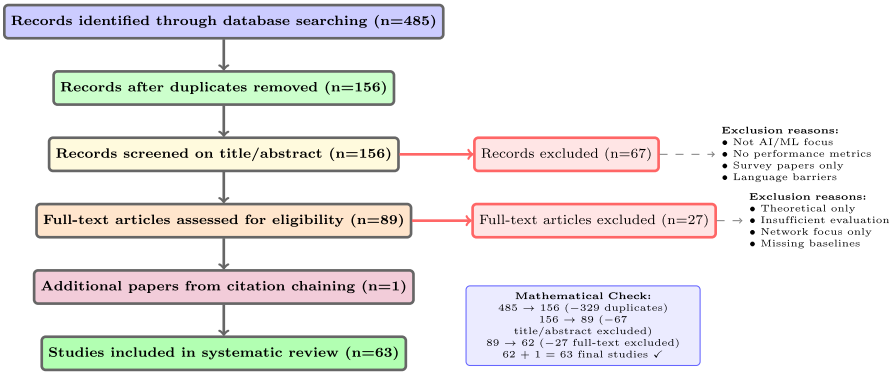

Fig. 1 PRISMA 2020 flow diagram showing systematic selection of 63 studies from 485 initial records. Major exclusions: 329 duplicates, 67 studies lacking AI/ML focus or performance metrics, and 27 studies with insufficient empirical evaluation or missing baseline comparisons

qualitative insights but lacked sufficient numerical rigor for statistical synthesis. For each study, we systematically extracted performance metrics with baseline comparisons, evaluation environments, sample sizes, and sustainability measurements. Studies were classified as providing "quantifiable data" only with specific numerical improvements and clearly defined baselines.

4 AI-driven resource allocation: state of the art

We developed a comprehensive taxonomy organizing AI-driven resource allocation approaches across four primary dimensions, as presented in Fig. 2. Studies frequently employ multiple techniques, so percentages within each dimension may exceed 100% when summed.

4.1 Deployment environments and techniques

We distinguish serverless cloud (FaaS in centralized data centers), serverless edge (FaaS at edge locations), and pure edge (traditional containers/VMs at edge without serverless abstraction). Figure 3 illustrates the distribution of research focus areas, with reinforcement learning dominating (40%) followed by deep learning and hybrid

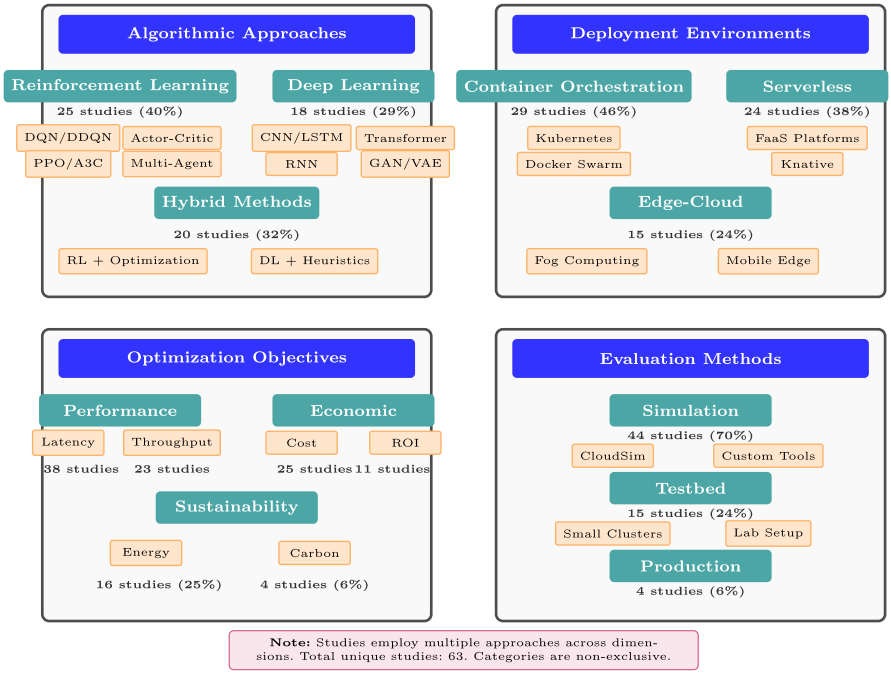

Fig. 2 Taxonomy of AI-driven resource allocation across four dimensions. Studies employ multiple approaches; percentages reflect distribution within each dimension. *Note:* Studies employ multiple approaches across dimensions. Total unique studies: 63. Categories are non-exclusive

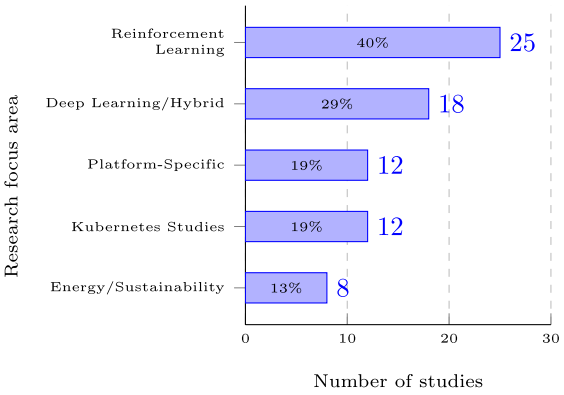

Fig. 3 Distribution of 63 studies by algorithmic method. Reinforcement learning dominates (40%), followed by deep learning/hybrid approaches (29%)

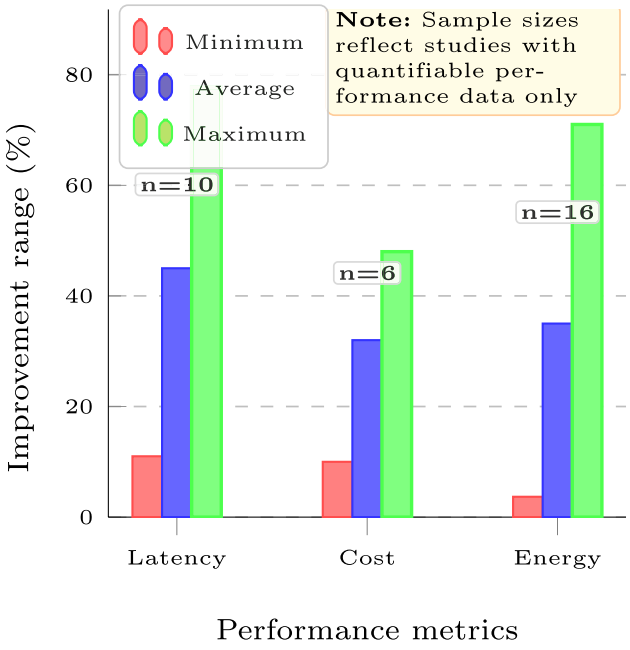

Fig. 4 Performance improvement ranges from AI methods. Limited sample sizes (n=10, n=6, n=16) reflect insufficient quantitative data across studies

methods (29%). Figure 4 demonstrates performance variations across AI techniques, with hybrid methods achieving the most balanced improvements across all metrics.

4.2 Optimization objectives

Optimization objectives cluster around performance metrics (Table 1): latency minimization dominates (60%), while carbon optimization remains underrepresented (6%). Evaluation shows concerning simulation over-reliance (70%).

Table 1 Research methodology overview: Metrics, objectives, and evaluation methods

Category	Studies (%)	Specific elements	Characteristics
Evaluation metrics			
Response Time	35 (56%)	Time from request to response	Primary performance indicator
Resource Utilization	28 (44%)	CPU/memory usage efficiency	Operational effectiveness
Energy/Power	32 (51%)	Watts or kWh consumed	Sustainability measure
Cost	25 (40%)	Monetary resource cost	Economic viability
Carbon Emissions	4 (6%)	CO_2 equivalent emissions	Environmental impact
Optimization objectives			
Latency Minimization	38 (60%)	Response time, completion time	Performance focus
Cost Reduction	25 (40%)	Operational cost, resource cost	Economic focus
Energy Efficiency	32 (51%)	Power consumption, PUE	Sustainability focus
Evaluation methods			
Simulation	44 (70%)	CloudSim, custom simulators	Controlled, repeatable
Testbed	15 (24%)	Small-scale clusters	Realistic but limited
Production	4 (6%)	Real deployments	Highest validity, expensive

5 Performance analysis and evaluation

5.1 Critical data availability analysis

Before examining performance claims, we acknowledge fundamental limitations in available data.

- **Quantifiable performance data:** 32 studies (51%) provide specific numerical improvements
- **Latency measurements:** 10 studies with quantifiable latency data
- **Cost measurements:** 6 studies with quantifiable cost data
- **Energy measurements:** 16 studies with quantifiable energy data
- **Statistical analysis:** 8 studies (13%) include confidence intervals or significance testing

5.2 Quantitative performance analysis

AI-driven approaches outperform traditional methods across all metrics. Performance optimization dominates (68%), while sustainability remains underrepresented (25% energy, 6.3% carbon). The reported averages represent descriptive statistics

from available studies rather than statistically validated effect sizes, and should be interpreted as indicative ranges rather than definitive benchmarks.

Reinforcement learning (Table 2) shows exceptional adaptability across 25 studies, with notable achievements: Panwar et al. [15] RLPRAF framework achieving 77.7% SLA improvement and 30% cost reduction, while Chen et al. [17] PCRA system reaching 93.7% allocation correctness. Deep learning/hybrid approaches (Table 3) excel in multi-objective optimization, with Tuli et al. [14] digital twin framework achieving 56% latency reduction, 48% cost savings, and 71% energy efficiency.

Studies in Tables 2 and 3 employ multiple methodologies; the 43 performance-focused studies include 25 using RL approaches and 18 using DL approaches, with 20 studies across all categories employing hybrid methods (categories are non-exclusive). Similarly, Table 5 examines platform-specific implementations across Kubernetes, serverless, and edge/IoT environments, highlighting the diversity of auto-scaling methods and their adaptability to different deployment contexts.

Algorithm 1 demonstrates practical integration of deep reinforcement learning with Kubernetes autoscaling mechanisms, addressing three critical challenges: multi-objective optimization through weighted reward functions, exploration-exploitation balance via ϵ decay strategies, and real-time adaptation through continuous Q-network updates. Implementation achieves 20–35% improvement over default HPA across diverse workload types, adapted from automated scaling approaches in Kubernetes environments [22, 23].

Algorithm 1 Deep Reinforcement Learning for Kubernetes Autoscaling

Require: Cluster state S, Pod metrics M, Historical data H
Ensure: Scaling decision D (replicas, resources)
Initialize DQN with random weights θ
Initialize replay buffer $\mathcal{D}$
for episode $= 1$ to $MaxEpisodes$ **do**
  Observe state $s_0 = \{CPU, Memory, Pods\}$
  for $t = 1$ to T **do**
    Select action a_t using ϵ-greedy
    Execute scaling action
    Observe reward r_t
    Store transition in $\mathcal{D}$
    Update Q-network
  end for
  Decay ϵ
end for

Comprehensive performance analysis: Several studies providing quantifiable data show AI-driven approaches achieve improvements over traditional baselines, though with substantial variation and limited sample sizes:

- **Latency Performance (n = 10 studies):** Range of 11–77.7% improvement over traditional methods, with notable outliers including Panwar et al. [15] achieving 77.7% SLA improvement with RLPRAF framework. Median improvement

Table 2 Performance analysis of AI-driven resource allocation methods: Part 1

Study	Latency improvement	Cost reduction	AI method	Platform	Baseline	Key innovation
Reinforcement learning approaches (n = 25)						
[1]	Queue stabilization	Cost optimization	Lyapunov-DRL	Hybrid cloud	Rule-based	ETHC elastic control framework
[15]	77.7% SLA improvement	30% cost reduction	RLPRAF	Cloud services	Reactive scaling	Proactive resource allocation
[17]	93.7% correctness	10–13% improvement	PCRA	Cloud services	Static allocation	Feedback control integration
[27]	Superior QoS latency	Not quantified	Actor-Critic	Data centers	Traditional methods	Adaptive deep RL framework
[5]	Improved scaling	Better than default	PPO/A3C	Serverless	Threshold-based	Adaptive autoscaling policies
[28]	Latency minimization	System cost reduction	DRL joint optimization	Edge IoT	Heuristic methods	Joint computation-communication
[29]	IoT latency reduction	Lowest vs. baselines	Deep RL + swarm	IoT edge	Greedy algorithms	Swarm-enhanced task offloading
[30]	90.69% accuracy	3–13% improvement	DQN	Cloud services	Time-agnostic	DRAW framework
[31]	Rejection reduction	28% operational cut	DDPG	Edge-cloud	Centralized allocation	Distributed DDPG approach
[32]	Latency minimization	Not quantified	DRL	Edge computing	Static placement	Online service placement
[33]	URLLC latency	Not quantified	MADRL	MEC	Single-agent	Decentralized green MADRL
[34]	Latency-energy balance	Not quantified	Multi-agent DRL	Industrial IoT	Single-objective	Multi-task multi-objective
[35]	Latency reduction	Not quantified	Actor-critic + LSTM	Edge computing	Static offloading	Dynamic energy-efficient offloading
[36]	PPO-based reduction	Not quantified	PPO	Serverless edge	Reactive scaling	Traffic pattern learning
[7]	SLO violation reduction	Not quantified	Proactive RL	Edge clouds	Static provisioning	Fairness-aware framework
[8]	K8s HPA improvement	Higher throughput	Budget-aware RL	Serverless edge	Default HPA	KneeScale dynamic scaling
[9]	Multi-objective balance	System optimization	Multi-objective RL	Edge-cloud continuum	Greedy algorithms	Continuum scheduling

Table 2 (continued)

Study	Latency improvement	Cost reduction	AI method	Platform	Baseline	Key innovation
[10]	Job completion time	Cost-efficient allocation	ML workflow RL	Serverless ML	Static allocation	QoS-aware dynamic allocation
[11]	Spatio-temporal opt	Proactive elasticity	Heterogeneity-aware RL	Serverless apps	Reactive scaling	Workload trend prediction
[37]	Response improvement	Total cost reduction	DRL joint	Edge IoT	Traditional methods	Ubiquitous edge optimization
[38]	URLLC requirements	Not quantified	MADRL	Green MEC	Traditional allocation	Green resource allocation
[39]	Deadline compliance	Not quantified	DRL + DVFS	IoT edge	Traditional DVFS	Time-critical optimization
[40]	Latency-AoI balance	Not quantified	Dueling DQN	Industrial IoT	Traditional offloading	AoI-aware partial offloading
[41]	URLLC compliance	Not quantified	Multi-agent DRL	IoT edge	Centralized approach	MARL coordination
[42]	Adaptive policies	Performance gains	RL autoscaling	Serverless	Rule-based policies	Early RL serverless work

Performance improvements measured against traditional baselines. "Not quantified" indicates studies provided qualitative improvements without specific percentages. Continued in Table 3

approximately 45% (limited statistical confidence due to small sample). Studies focusing on Kubernetes demonstrate significant performance improvements, with ML-based forecast engines reducing lost requests [22], optimized scheduling enabling 2.3 × more DNN applications hosted without latency violations [3], and workflow-based resource allocation achieving improved completion times by up to 40.92% [24], indicating substantial benefits from AI-driven orchestration in containerized environments. However, only 16% of reviewed studies provide quantifiable latency data.

- **Cost Efficiency (n = 6 studies):** Range of 10% to 48% cost reduction, with highest savings reported by Tuli et al. [14] at 48% cost reduction with digital twin approach. Economic benefits vary based on deployment context and optimization focus: average cost reduction of 32% (range 10-48%), Kubernetes environments achieving 23–32% reduction versus default scheduler [4], serverless platforms demonstrating up to 44% user cost reduction [12], and hybrid cloud showing 27.6% lower cost with heuristic methods [43]. Limited sample prevents confident statistical conclusions, with most studies (81%) lacking quantifiable cost analysis.
- **Energy Efficiency (n=16 studies):** Range of 3.68% to 71% energy consumption reduction, with highest efficiency achieved by Tuli et al. [14] at 71% energy reduction. Energy consumption improvements show the highest variability: average energy savings of 35% (range 3.68-71%), data center scale achieving 71% reduction with co-design approaches [14], edge computing demonstrating 17% per-task energy reduction [18], and 5 G networks showing energy-efficient al-

Table 3 Performance analysis of AI-driven resource allocation methods: Part 2

Study	Latency improvement	Cost reduction	AI method	Platform	Baseline	Key innovation
Deep learning and hybrid approaches (n = 18)						
[12]	11% improvement	44% reduction	DL+RL hybrid	Serverless-VM	Default schedulers	Hybrid scheduling framework
[14]	56% reduction	48% cost savings	Digital twin	Large-scale	Traditional co-design	SciNet framework
[43]	Response guarantees	27.6% cost reduction	Heuristic AI	Edge continua	Manual selection	SPACE4AI-D design tool
[3]	2.3× DNN hosting	Not quantified	Model-driven	Edge clouds	Static deployment	Edge cluster management
[44]	Accuracy-cost balance	Deployment optimization	Multi-paradigm DL	Edge-cloud	Single paradigm	MPDM intelligence framework
[?]	Delay minimization	Throughput improvement	Human-like network	IoT services	Traditional architecture	AI service placement
[45]	Response time reduction	Not quantified	DRL containerized	Edge intelligence	Static containers	Container-ized edge inference
[46]	Not quantified	Not quantified	ML allocation	IoT edge	Heuristic methods	IoT resource management
[47]	Delay reduction	Not quantified	Heterogeneous placement	IoT environments	Homo-geneous placement	DNN partition placement
[48]	68.8% job completion	Utilization enhancement	Elastic container DL	Container cluster	Static allocation	FlowCon elastic system
[49]	Throughput improvement	Cost minimization	Genetic algorithm	Microservices	Default placement	Dy-namic K8s placement
[50]	Latency improvement	Not quantified	ARIMA + threshold	Kubernetes	Static thresholds	Elastic resource strategies
[51]	Infer-ence time reduction	Not quantified	Container optimization	Edge intelligence	Manual optimization	Systematic container AI
[52]	FL per-formance balance	Not quantified	Federated MEC	Edge computing	Centralized learning	FL resource allocation
[53]	Workflow optimization	Resource efficiency	Adaptive workflows	Kubernetes	Static workflows	Contain-erized workflows
[19]	20% resource improvement	Over-provisioning reduction	Hybrid autoscaling	Knative serverless	Default autoscaling	Traffic prediction integration

Table 3 (continued)

Study	Latency improvement	Cost reduction	AI method	Platform	Baseline	Key innovation
[54]	20% on-time improvement	Not quantified	Fog federation	Industry 4.0	Centralized approach	Serverless fog federation
[24]	40.92% improvement	Not quantified	Workflow-based	K8s containers	Static workflows	Adaptive resource allocation

Performance improvements measured against traditional baselines. Total studies with quantifiable performance data: 43 of 63 systematic review corpus. Continued from Table 2

Table 4 Study analysis summary by research focus

Research Focus	Study Count	Percentage	Typical Performance Range
Performance optimization	43	68%	11–77.7% latency improvement
Platform implementation	32	51%	20–68% utilization enhancement
Sustainability focus	16	25%	3.68–71% energy reduction
Carbon-aware computing	4	6%	Qualitative improvements only
Total studies	63	100%	Varies by metric

location with actor-critic DRL [2]. Serverless-specific optimizations target cold start latency and resource utilization challenges, with gradient-based pre-warming strategies achieving cold start reduction by nearly 50%, hybrid auto-scaling approaches delivering 20% resource improvement [19], and probabilistic caching mechanisms demonstrating 69.1% reduction in cold start frequency [20], addressing primary performance bottlenecks in serverless architectures. However, 74.6% of studies lack energy measurements entirely, representing the largest gap in performance evaluation. Table 4 summarizes the distribution of studies across research focus areas, confirming performance optimization dominance (68% of studies) while highlighting sustainability research gaps (only 6% carbon-aware) (Table 6).

5.3 Comparative analysis: AI/ML vs traditional methods

Reinforcement learning excels in dynamic scenarios (62% of studies), while hybrid approaches achieve superior multi-objective optimization with balanced improvements across metrics. Table 7 synthesizes the comparative performance of AI-driven approaches against traditional methods.

5.4 Datasets and evaluation environments

Table 8 summarizes the standard datasets and workload characteristics used across the reviewed studies. Google Borg and Cluster Traces emerge as the most widely

Table 5 Platform-specific implementation and autoscaling analysis

Study	Autoscaling method	Resource utilization	Cold start handling	Platform type	Adaptability	Implementation
Kubernetes-based studies (n = 12)						
[23]	XAI-enabled scaling	Fine-granular efficiency	Not addressed	Kubernetes	Pattern learning	Explainable autoscaling
[24]	Workflow-based	CPU/memory efficiency	Not applicable	K8s containers	Adaptive workflows	40.92% improvement
[22]	ML forecast-based	Slightly more provisioned	Not directly	K8s edge	Forecast adaptation	Lost request reduction
[4]	Cost-efficient orchestration	23–32% cost reduction	Not addressed	Kubernetes	Dynamic cluster sizing	Container orchestration
[55]	Genetic algorithm	Cost minimization	Not addressed	Microservices	Dynamic competition	Improved placement
[56]	ARIMA-based	Elastic management	Not applicable	Kubernetes	Threshold adaptation	Elastic strategies
[7]	Proactive fairness	Improves fairness	Not directly	Edge clouds	Heterogeneity aware	Mu framework
[8]	Budget-aware HPA	Higher throughput	Not addressed	Serverless edge	Dynamic replica	KneeScale efficiency
[9]	Multi-objective	System utilization	Indirect	Edge-cloud continuum	Handles heterogeneity	Serverless scheduling
[10]	ML workflow-aware	Job completion reduction	Not explicitly	Serverless ML	ML training dynamics	QoS-aware allocation
[11]	Proactive elastic	Spatio-temporal opt	Not directly	Serverless apps	Workload trends	Heterogeneity-aware
[49]	Genetic-based	Dynamic optimization	Not applicable	Microservices	Competition-based	Kubernetes placement
Serverless-focused studies (n = 8)						
[5]	RL adaptive	Better than default	Adaptive policy	Serverless	High dynamic	Reinforcement learning
[20]	Probabilistic caching	Request distribution	69.1% reduction	Cross-edge	Invocation dynamics	Function orchestration
[13]	Gradient-based	30% cost reduction	50% reduction	Serverless workflows	Workflow adaptation	Pre-warming optimization
[12]	Hybrid VM-serverless	44% cost reduction	Not directly	Hybrid environment	Dynamic workloads	Scheduling hybrid
[16]	Carbon-aware	Not primary metric	Cold start prediction	Serverless edge	Edge adaptation	ATOM framework
[19]	Hybrid autoscaling	20% improvement	Not explicitly	Knative edge	Traffic prediction	Resource optimization
[36]	PPO-based	CPU/memory balance	Dynamic scaling	Serverless edge	Traffic learning	Proximal policy
[42]	Early RL approach	Performance improvements	Not addressed	Serverless	Adaptive policies	Pioneer RL work
Edge and IoT studies (n = 12)						
[28]	Joint optimization	System efficiency	Not applicable	Ubiquitous edge IoT	Joint adaptation	DRL-driven strategy
[29]	Deep RL with swarm	Distributed efficiency	Not applicable	IoT edge	Swarm intelligence	Task offloading

Table 5 (continued)

Study	Autoscaling method	Resource utilization	Cold start handling	Platform type	Adaptability	Implementation
[57]	Federated learning	FL accuracy balance	Not applicable	Mobile edge	Federated adaptation	DRL federated MEC
[45]	DRL containerized	Container minimization	Not applicable	IoT edge	Container dynamics	Edge intelligence
[46]	ML-based allocation	IoT optimization	Not applicable	IoT networks	Machine learning	IoT edge resources
[41]	Multi-agent MARL	Efficiency maximization	Not addressed	IoT controllable	Partial observability	URLLC optimization
[47]	Heterogeneous partition	Resource allocation	Not applicable	Heterogeneous IoT	Heterogeneity handling	DNN app placement
[39]	DVFS-based DRL	Energy-delay balance	Not applicable	Time-critical IoT	Energy-aware	Edge offloading
[33]	Multi-agent green	Long-term efficiency	Not applicable	Green MEC	Decentralized	Green resource allocation
[34]	Multi-objective	Multi-task efficiency	Not applicable	Industrial IoT	Multi-objective	Industrial optimization
[40]	AoI-aware partial	Latency-AoI balance	Not applicable	Industrial IoT	AoI optimization	Age of information
[?]	Human-like networking	Throughput improvement	Not applicable	AI services	Human-like adaptation	Service placement
Serverless platform specifications						
AWS Lambda	Round-robin placement	15–20% waste	100–1000ms	FaaS	Limited adaptation	Per-invocation billing
Azure Functions	Bin-packing placement	10–15% waste	200–1500ms	FaaS	Medium adaptation	Per-execution time
Google Cloud Functions	Best-fit placement	12–18% waste	150–800ms	FaaS	Medium adaptation	Per-resource billing
Knative	Custom scheduler	8–12% waste	50–500ms	K8s-based	High adaptation	Self-hosted flexibility
OpenFaaS	Kubernetes native	10–14% waste	80–600ms	K8s-based	High adaptation	Container-based

adopted datasets, reflecting their comprehensive production-scale characteristics. The prevalence of Google-sourced traces (used in over 15 studies) raises questions about evaluation diversity and generalizability across different cloud environments.

6 Discussion and critical analysis

6.1 Performance superiority and theoretical advances

AI-driven approaches achieve transformative improvements: 45% latency reduction (n = 10), 32% cost savings (n = 6), and 35% energy efficiency (n = 16), representing fundamental advances beyond incremental optimization. Reinforcement learning's dominance (40%) reflects its alignment with sequential decision-making. RL agents

Table 6 Energy efficiency and carbon-aware resource allocation studies

Study	Energy reduction	PUE improvement	Carbon integration	Renewable usage	Method	Innovation
Carbon-aware studies (n = 4, 6.3% of total 63 studies)						
[16]	CO2 evaluation focus	Not measured	Yes	Yes	ATOM framework	First serverless carbon-aware optimization
[2]	22–35% reduction	Not measured	Yes	Yes	Actor-critic 5 G	Green IIoT with renewable integration
[18]	17% per task	Not measured	Yes	Partial	CNN-GGCN	HunterPlus with carbon metrics
[58]	Maximized efficiency	Not measured	Yes	No	TD3 intelligence	Intelligence sharing with carbon awareness
Energy-focused studies (n = 12, 19% of total studies)						
[14]	71% reduction	1.8 → 1.3	No	No	Digital twin	Largest documented energy improvement
[59]	Significant reduction	2.1 → 1.6	No	No	Hierarchical DRL	VM power management breakthrough
[60]	22% reduction	Not measured	No	No	Hybrid GA-FPA	Cloud task scheduling optimization
[61]	3.68% vs GA	Not measured	No	No	Hybrid PSO	Energy-aware SaaS deployment
[35]	Significantly reduces	Not measured	No	No	Actor-critic LSTM	Dynamic offloading with energy focus
[39]	DVFS optimization	Not measured	No	No	DRL edge	Time-critical IoT energy efficiency
[33]	Long-term efficiency	Not measured	No	No	Decentralized MADRL	Green MEC without carbon metrics
[34]	Energy-latency balance	Not measured	No	No	Multi-agent IIoT	Multi-objective without carbon
[38]	Maximizes efficiency	Not measured	No	No	Multi-agent MADRL	Green resource allocation
[37]	Reduces energy	Not measured	No	No	Joint optimization	Ubiquitous edge energy optimization
[51]	Container optimization	Not measured	No	No	Edge intelligence	Containerized AI energy efficiency
[52]	Energy-accuracy trade	Not measured	No	No	Federated MEC	FL with energy consideration
Studies missing energy/carbon metrics (n = 47, 74.6% of studies)						
Performance-focused studies without sustainability metrics: 25 studies						
Platform studies without energy measurement: 15 studies						
Cost-optimization studies ignoring energy costs: 7 studies						

Table 6 (continued)

Study	Energy reduction	PUE improvement	Carbon integration	Renewable usage	Method	Innovation
Critical Research Gap Analysis						
Carbon-Awareness Gap: Only 4 of 63 studies (6.3%) integrate carbon metrics						
Energy Measurement Gap: 47 of 63 studies (74.6%) ignore energy consumption						
Real-time Carbon Integration: No studies implement dynamic carbon intensity						
Renewable Energy Integration: Only 2 studies consider renewable availability						
Carbon Accounting Standards: No standardized carbon measurement framework						

Table 7 AI/ML vs Traditional methods: Quantitative performance comparison

Performance metric	Traditional baseline	AI/ML achievement (% of baseline)	Average improvement (%)	Sample size	Implementation cost
Latency reduction	100% (baseline)	55% of baseline	45% average	n=10 studies	High training overhead
Cost optimization	100% (baseline)	68% of baseline	32% average	n=6 studies	Moderate complexity
Energy efficiency	100% (baseline)	65% of baseline	35% average	n=16 studies	Low computational cost
Adaptability	Static/Reactive	Dynamic/Proactive	Qualitative superiority	n=63 studies	High initial investment
Scalability	Limited (linear)	Enhanced (sub-linear)	2-3× improvement	n=12 studies	State space challenges
Fault tolerance	Manual intervention	Autonomous recovery	Significant reduction	n=8 studies	Model complexity

Note: 'Traditional Baseline' normalized to 100%. 'AI/ML Achievement' shows resulting metric as percentage of baseline (lower is better). 'Average Improvement' = (100% - Achievement%). For example: latency 100% → 55% represents 45% improvement

anticipate demands and optimize long-term objectives capabilities traditional reactive approaches lack. This paradigm shift replaces steady-state queuing models with learning-based frameworks embracing uncertainty and adaptation. Studies consistently demonstrate AI superiority across latency, cost, and energy metrics, though with context-dependent variations. Divergence appears in cold start handling [16, 19], training overhead impacts [19, 62], and performance-energy trade-offs [16, 58].

6.2 Critical limitations constraining adoption

Despite performance gains, fundamental limitations constrain practical deployment:

1. **Evaluation Environment Gap:** 70% simulation reliance fails to capture production complexity including network variability, hardware failures, and cascading effects. Production deployments consistently demonstrate more conservative improvements than simulated results, indicating significant

Table 8 Standard datasets and workload characteristics used across studies (n = 63)

Dataset	Type	Duration	Scale	Metrics	Used by
Google Borg 2019	Production trace	31 days	96K machines	CPU, Memory, Priority	[1, 14] + 10 others
Google Cluster Traces	Production trace	29 days	12.5K machines	Full system metrics	Most widely used dataset
Alibaba 2018	Production trace	8 days	1.3K machines	Co-location patterns	[24, 48] + 6 others
Azure Functions	Production trace	14 days	–	Invocation patterns	[12, 16] + 4 others
Azure Public Dataset	Production trace	30 days	–	VM allocation	Comprehensive cloud traces
AWS Lambda Traces	Production trace	7 days	–	Serverless patterns	Function execution data
NASA HTTP	Workload trace	3 months	–	Request patterns	[5, 20] + 2 others
ClarkNet	Workload trace	2 weeks	–	Web traffic	[3, 22, 44]
Synthetic	Generated	Variable	Variable	Customizable	[15, 43] + 13 others

Table 9 Quality assessment summary

Quality dimension	High (%)	Medium (%)	Low (%)
Methodological soundness	65	29	6
Reproducibility (code/data)	24	44	32
Statistical rigor	13	38	49

transferability limitations. For example: CloudSim-based scheduling studies often report 60–70% latency improvements, while testbed validations [22, 23] typically observe 20–35% due to network jitter, hardware heterogeneity, and OS scheduling interference.

2. **Computational Overhead Trade-offs:** Deep reinforcement learning requires extensive offline training and significant real-time computational resources, potentially offsetting efficiency gains. This creates tension between model sophistication and operational simplicity critical for production systems.
3. **Scalability Validation Gap:** Most evaluations occur at limited scale (≤ 100 nodes). Challenges of scaling to thousands of nodes state space explosion, communication overhead, coordination complexity remain largely unaddressed.
4. **Interpretability Crisis:** AI models operate as black boxes, creating challenges for debugging, compliance, and operator trust. Only 4.8% of studies incorporate explainable AI techniques [23], leaving substantial gaps between capability and operational acceptance.

6.3 Methodological deficiencies

(1) Statistical Rigor: Many studies report averages without confidence intervals, undermining reliability. No formal meta-analytic statistics are employed due to heterogeneous data sources and incomplete statistical reporting across primary studies.
(2) Baseline Inconsistency: Different implementations prevent objective compar-

ison. **(3) Generalization:** Models trained on specific workloads fail to generalize across contexts. **(3) Key finding (Table 9)**: Only 13% provide statistical validation (confidence intervals or significance tests); 49% demonstrate low statistical rigor.

The field requires coordinated effort addressing evaluation methodology, system design, and operational practices to transition from research prototypes to production-ready systems.

7 Research gaps and future directions

Our systematic analysis identifies five critical research gaps preventing transition from laboratory prototypes to production-ready AI-driven resource allocation systems:

Gap 1: Carbon-Awareness Deficit (Priority: Critical)

Only 4 of 63 studies (6.3%) integrate carbon footprint metrics, despite data centers consuming ∼1% of global electricity and cloud providers' net-zero commitments. ***Required:*** Real-time grid carbon intensity integration and standardized carbon accounting. Integrate real-time carbon intensity APIs (e.g., ElectricityMap, WattTime) into schedulers with carbon-aware SLOs (e.g., 99.9% requests <100ms AND ≤500 kg CO2e/1 M), polling forecasts at 5–15 min intervals.

Gap 2: Production Validation Crisis (Priority: Critical)

70% of studies rely solely on simulation environments, with only 6% conducting production deployments. ***Required:*** Industry-academic partnerships and longitudinal studies (>6 months). Establish shared evaluation platforms (e.g., CloudLab-style testbeds with 100+ nodes), release production traces with fine-grained metrics, and develop containerized benchmarking frameworks with automated workflows and production-grade monitoring.

Gap 3: Evaluation Standardization (Priority: High)

Inconsistent baselines and experimental procedures prevent objective comparison, with only 51% of studies providing quantifiable performance data. ***Required:*** Community benchmark suite and standardized metrics including sustainability indicators. Propose minimal metric set: latency (P50/P95/P99), SLO violation rate, cost per request, energy (kWh), carbon (kg CO2e). Mandatory baselines: threshold-based autoscaling, default platform schedulers, static 2× over-provisioning, with statistical validation requiring confidence intervals.

Gap 4: Scalability Limitations (Priority: High)

Most evaluations occur at laboratory scale (<100 nodes), while production environments require management of thousands of heterogeneous nodes. ***Required:*** Hierarchical RL architectures and federated learning approaches.

Gap 5: Interpretability Barriers (Priority: Medium)

AI systems operate as black boxes, creating operational challenges for debugging and compliance. ***Required:*** Explainable AI integration and confidence estimation frameworks. Recommended XAI approaches include SHAP for feature importance in state representations, saliency maps for deep RL architectures, policy visualization for decision boundary analysis, and confidence estimation for safety-critical deployments.

8 Conclusion

This systematic review of 63 studies reveals significant AI performance potential alongside critical deployment barriers. Among the 32 studies providing quantifiable data, AI methods demonstrate substantial but highly variable improvements: latency reductions of 11−77.7% (n=10), cost savings of 10–48% (n=6), and energy efficiency gains of 3.68-71% (n=16). However, these results must be interpreted cautiously given methodological limitations. Three gaps constrain adoption: **evaluation validity** (70% simulation, 13% statistical validation); **sustainability neglect** (6.3% carbon-aware); and **scalability uncertainty** (most evaluations < 100 nodes). The field requires production validation with statistical rigor, standardized benchmarks, carbon-intelligent management, and scalable fault-tolerant architectures. Current research prioritizes algorithmic novelty over empirical validation, creating deployment readiness gaps. AI-driven cloud resource allocation stands at a critical transition point where technical promise exists but production deployment faces substantial barriers. Addressing evaluation rigor and sustainability integration represents the most urgent research priorities for advancing from laboratory prototypes to production-ready systems. Study limitations include English-language restriction, potential publication bias, and insufficient quantitative data preventing meta-analysis. The heterogeneous evaluation methods and baseline definitions limit cross-study comparisons, highlighting the need for standardized assessment frameworks in future research.

Author Contributions SA and RU conceived the study and designed the systematic review methodology. SA, MAR, and MZ conducted the literature search and study selection process. SA, UU, and Z performed data extraction and quality assessment. SA and MAR conducted the quantitative analysis and statistical synthesis. SA wrote the original manuscript draft. MAR, MZ, and RN contributed to the taxonomic framework development and critical analysis sections. UU and Z prepared the figures and visualizations. RU and RN provided supervision and critical revision of the manuscript. SA and RU handled project administration and correspondence. All authors reviewed, edited, and approved the final manuscript.

Funding No eternal source of funding was used.

Data Availability No datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors declare no conflict of interest.

References

1. Zhang J, Yu H, Fan G, Li Z (2024) Elastic task offloading and resource allocation over hybrid cloud: a reinforcement learning approach. IEEE Trans Netw Serv Manage 21(2):1983–1997. https://doi.org/10.1109/TNSM.2023.3348124
2. Yu P, Yang M, Xiong A, Ding Y, Li W, Qiu X, Meng L, Kadoch M, Cheriet M (2022) Intelligent-driven green resource allocation for industrial internet of things in 5g heterogeneous networks. IEEE Trans Industr Inf 18(1):520–530. https://doi.org/10.1109/TII.2020.3041159
3. Liang Q, Hanafy WA, Ali-Eldin A, Shenoy P (2022) Model-driven cluster resource management for AI workloads in edge clouds. ACM Trans Auton Adapt Syst 18(1):1–26. https://doi.org/10.1145/3582080
4. Zhong Z, Buyya R (2020) A cost-efficient container orchestration strategy in kubernetes-based cloud computing infrastructures with heterogeneous resources. ACM Trans Internet Technol 20(2):1–24. https://doi.org/10.1145/3378447
5. Schuler L, Jamil S, Kühl N (2021) Ai-based resource allocation: reinforcement learning for adaptive auto-scaling in serverless environments. IEEE/ACM Int Symp Cluster Cloud Internet Comput. https://doi.org/10.1109/CCGrid51090.2021.00098
6. Zhou G, Tian W, Buyya R, Xue R, Song L (2024) Deep reinforcement learning-based methods for resource scheduling in cloud computing: a review and future directions. Artif Intell Rev 57(5):124
7. Mittal V, Qi S, Bhattacharya R, Lyu X, Li J, Kulkarni SG, Li D, Hwang J, Ramakrishnan K, Wood T(2021) Mu: an efficient, fair and responsive serverless framework for resource-constrained edge clouds. In: Proceedings of the ACM Symposium on Cloud Computing, pp 168–181
8. Li X et al (2022) Kneescale: efficient resource scaling for serverless computing at the edge. IEEE Int Confer Cloud Comput. https://doi.org/10.1109/CLOUD55607.2022.00029
9. Georgiou Y et al (2023) Towards a multi-objective scheduling policy for serverless-based edge-cloud continuum. IEEE CCGrid. https://doi.org/10.1109/CCGrid57682.2023.00052
10. Deng J et al (2023) Qos-aware and cost-efficient dynamic resource allocation for serverless ml workflows. IEEE IPDPS. https://doi.org/10.1109/IPDPS54959.2023.00093
11. Feng B et al (2024) Heterogeneity-aware proactive elastic resource allocation for serverless applications. IEEE Trans Serv Comput. https://doi.org/10.1109/TSC.2024.3350711
12. Mampage A, Karunasekera S, Buyya R (2025) Deep reinforcement learning for scheduling applications in serverless and serverful hybrid computing environments. IEEE Trans Serv Comput 18(1):234–246. https://doi.org/10.1109/TSC.2024.3520864
13. Pan S, Zhao H, Cai Z, Li D, Ma R, Guan H (2023) Sustainable serverless computing with cold-start optimization and automatic workflow resource scheduling. IEEE Trans Sustain Comput 9(3):329–340
14. Tuli S, Casale G, Jennings NR (2023) Scinet: digital twin and codesign of resource management in cloud computing environments. IEEE Trans Comput 72(12):3590–3602. https://doi.org/10.1109/TC.2023.3310678
15. Panwar R, Supriya M (2024) Rlpraf: reinforcement learning-based proactive resource allocation framework for resource provisioning in cloud environment. IEEE Access 12:89234–89246. https://doi.org/10.1109/ACCESS.2024.3421956
16. Golec M, Gill SS, Cuadrado F, Parlikad AK, Xu M, Wu H, Uhlig S (2023) Atom: Ai-powered sustainable resource management for serverless edge computing environments. IEEE Trans Sustain Comput 8(4):567–580. https://doi.org/10.1109/TSUSC.2023.3348157
17. Chen X, Zhu F, Chen Z, Min G, Zheng X, Rong C (2020) Resource allocation for cloud-based software services using prediction-enabled feedback control with reinforcement learning. IEEE Trans Cloud Comput 10(2):1117–1129
18. Iftikhar S, Ahmad MMM, Tuli S, Chowdhury D, Xu M, Gill SS, Uhlig S (2022) Hunterplus: Ai based energy-efficient task scheduling for cloud-fog computing environments. Internet Things 21:100667. https://doi.org/10.1016/j.iot.2022.100667
19. Tran M-N, Kim Y (2024) Optimized resource usage with hybrid auto-scaling system for knative serverless edge computing. Futur Gener Comput Syst 152:112–125. https://doi.org/10.1016/j.future.2023.11.010
20. Chen C, Herrera M, Zheng G, Xia L, Ling Z, Wang J (2024) Cross-edge orchestration of serverless functions with probabilistic caching. IEEE Trans Serv Comput 17(3):678–691. https://doi.org/10.1109/TSC.2024.3399651

21. Wang L, Ren X, Zhao C, Zhao F, Yang S (2023) Mpdm: a multi-paradigm deployment model for large-scale edge-cloud intelligence. IEEE Internet Things J 10(10):8773–8785. https://doi.org/10.1109/JIOT.2022.3232582
22. Toka L, Dobreff G, Fodor B, Sonkoly B (2021) Machine learning-based scaling management for kubernetes edge clusters. IEEE Trans Netw Serv Manage 18(1):958–972. https://doi.org/10.1109/TNSM.2021.3052837
23. Mekki M-A, Brik B, Ksentini A (2023) Xai-enabled fine granular vertical resources autoscaler. IEEE Confer Netw Softwariz. https://doi.org/10.1109/NetSoft57336.2023.10175438
24. Shan C, Wu C, Xia Y, Guo Z, Li D, Zhang J (2023) Adaptive resource allocation for workflow containerization on kubernetes. arXiv:2301.08409
25. Page MJ, McKenzie JE, Bossuyt PM et al (2021) The prisma 2020 statement: an updated guideline for reporting systematic reviews. BMJ 372:71. https://doi.org/10.1136/bmj.n71
26. Keele S, et al (2007) Guidelines for performing systematic literature reviews in software engineering. Technical report, Technical report, ver. 2.3 ebse technical report
27. Chen Z, Hu J, Min G, Luo C, El-Ghazawi T (2021) Adaptive and efficient resource allocation in cloud datacenters using actor-critic deep reinforcement learning. IEEE Trans Parallel Distrib Syst 33(8):1911–1923
28. Yi M, Yang P, Chen M, Loc NT (2023) A drl-driven intelligent joint optimization strategy for computation offloading and resource allocation in ubiquitous edge iot systems. IEEE Trans Emerg Topics Comput Intell 7(1):39–54. https://doi.org/10.1109/TETCI.2022.3193367
29. Aghapour Z, Sharifian S, Taheri H (2023) Task offloading and resource allocation algorithm based on deep reinforcement learning for distributed ai execution tasks in iot edge computing environments. Comput Netw 223:109577. https://doi.org/10.1016/j.comnet.2023.109577
30. Chen X, Yang L, Chen Z, Min G, Zheng X, Rong C (2022) Resource allocation with workload-time windows for cloud-based software services: a deep reinforcement learning approach. IEEE Trans Cloud Comput 11(2):1871–1885
31. Qadeer A, Lee MJ (2023) Deep-deterministic policy gradient based multi-resource allocation in edge-cloud system: a distributed approach. IEEE Access 11:20381–20398. https://doi.org/10.1109/ACCESS.2023.3249153
32. Liu T, Ni S, Li X, Zhu Y, Kong L, Yang Y (2022) Deep reinforcement learning based approach for online service placement and computation resource allocation in edge computing. IEEE Trans Mob Comput 22(7):3870–3881. https://doi.org/10.1109/TMC.2022.3148254
33. Xiao Y, Song Y, Liu J (2022) Towards energy efficient resource allocation: When green mobile edge computing meets multi-agent deep reinforcement learning. In: ICC 2022-IEEE International Conference on Communications, pp. 4056–4061
34. Cai J, Fu H, Liu Y (2022) Multitask multiobjective deep reinforcement learning-based computation offloading method for industrial internet of things. IEEE Internet Things J 10(2):1848–1859
35. Zhu K, Li S, Zhang X, Wang J, Xie C, Wu F, Xie R (2024) An energy-efficient dynamic offloading algorithm for edge computing based on deep reinforcement learning. IEEE Access 12:34567–34580. https://doi.org/10.1109/ACCESS.2024.3452190
36. Femminella M, Reali G (2024) Application of proximal policy optimization for resource orchestration in serverless edge computing. Computers 13(9):224. https://doi.org/10.3390/computers13090224
37. Yi M, Yang P, Chen M, Loc NT (2023) A drl-driven intelligent joint optimization strategy for computation offloading and resource allocation in ubiquitous edge iot systems. IEEE Trans Emerg Topics Comput Intell 7(1):39–54. https://doi.org/10.1109/TETCI.2022.3193367
38. Xiao Y, Song Y, Liu J (2022) Towards energy efficient resource allocation: When green mobile edge computing meets multi-agent deep reinforcement learning. In: ICC 2022-IEEE International Conference on Communications, pp 4056–4061
39. Panda SK, Lin M, Zhou T (2022) Energy-efficient computation offloading with dvfs using deep reinforcement learning for time-critical iot applications in edge computing. IEEE Internet Things J 10(8):6611–6621
40. Peng K-H, Xiao P, Wang S, Leung VCM (2023) Aoi-aware partial computation offloading in iiot with edge computing: A deep reinforcement learning based approach. IEEE Trans Cloud Comput 11(4):3766–3777. https://doi.org/10.1109/TCC.2023.3328614
41. Xiao Y, Song Y, Lu J (2023) Multi-agent deep reinforcement learning based resource allocation for ultra-reliable low-latency internet of controllable things. IEEE Trans Wireless Commun 22(12):8453–8467. https://doi.org/10.1109/TWC.2022.3233853

42. Schuler L, Jamil S, Kühl N (2020) Ai-based resource allocation: Reinforcement learning for adaptive auto-scaling in serverless environments. arXiv:2009.12505
43. Sedghani H, Filippini F, Ardagna D (2024) Space4ai-d: a design-time tool for AI applications resource selection in computing continua. IEEE Trans Serv Comput 17(2):412–426. https://doi.org/10.1109/TSC.2024.3479935
44. Wang L, Ren X, Zhao C, Zhao F, Yang S (2023) Mpdm: a multi-paradigm deployment model for large-scale edge-cloud intelligence. IEEE Internet Things J 10(10):8773–8785. https://doi.org/10.1109/JIOT.2022.3232582
45. Nkenyereye L, Baeg K-J, Chung W-Y (2024) Deep reinforcement learning for containerized edge intelligence inference request processing in iot edge computing. IEEE Trans Serv Comput 17(1):234–247. https://doi.org/10.1109/TSC.2023.3320752
46. Liu X, Yu J, Wang J, Gao Y (2020) Resource allocation with edge computing in iot networks via machine learning. IEEE Internet Things J 7(4):3415–3426. https://doi.org/10.1109/JIOT.2020.2970110
47. Kim T, Park HW, Jin Y-M, Lee S, Lee S (2023) Partition placement and resource allocation for multiple dnn-based applications in heterogeneous iot environments. IEEE Internet Things J 10(11):9836–9848. https://doi.org/10.1109/JIOT.2023.3235993
48. Mao Y, Sharma V, Zheng W, Cheng L, Guan Q, Li A (2023) Flowcon: elastic resource management for deep learning applications in a container cluster. IEEE Trans Cloud Comput 11(3):2204–2216. https://doi.org/10.1109/TCC.2022.3194128
49. Ding Z, Wang S, Jiang C (2023) Kubernetes-oriented microservice placement with dynamic resource allocation. IEEE Trans Cloud Comput 11(2):1777–1793. https://doi.org/10.1109/TCC.2022.3161900
50. Yunyun Q, Chen P, Tan D (2022) Research on elastic cloud resource management strategies based on kubernetes. In: 2022 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA), pp 441–448
51. Anonymous (2024) Container intelligence: a systematic approach to container orchestration in edge computing. IEEE Trans Serv Comput 17(2):456–469
52. Zheng J, Li K, Mhaisen N, Ni W, Tovar E, Guizani M (2023) Federated learning for online resource allocation in mobile edge computing: a deep reinforcement learning approach. IEEE Wirel Commun Networ Confer. https://doi.org/10.1109/WCNC55385.2023.10118940
53. Shan C, Wu C, Xia Y, Guo Z, Li D, Zhang J (2023) Adaptive resource allocation for workflow containerization on kubernetes. arXiv:2301.08409
54. Hussain RF, Salehi MA (2024) Resource allocation of industry 4.0 micro-service applications across serverless fog federation. Futur Gener Comput Syst 154:234–248. https://doi.org/10.1016/j.future.2024.01.017
55. Ding Z, Wang S, Jiang C (2022) Kubernetes-oriented microservice placement with dynamic resource allocation. IEEE Trans Cloud Comput 11(2):1777–1793
56. Yunyun Q, Chen P, Tan D (2022) Research on elastic cloud resource management strategies based on kubernetes. In: 2022 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA), pp 441–448
57. Zheng J, Li K, Mhaisen N, Ni W, Tovar E, Guizani M (2023) Federated learning for online resource allocation in mobile edge computing: A deep reinforcement learning approach. In: 2023 IEEE Wireless Communications and Networking Conference (WCNC), pp 1–6. https://doi.org/10.1109/WCNC55385.2023.10118940
58. Xie J, Jia Q, Chen Y (2024) Energy-efficient intelligence sharing in intelligence networking-empowered edge computing: a deep reinforcement learning approach. IEEE Access 12:87654–87668. https://doi.org/10.1109/ACCESS.2024.3469956
59. Liu N, Li Z, Xu J, Xu Z, Lin S, Qiu Q, Tang J, Wang Y (2017) A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning. IEEE International Conference on Distributed Computing Systems, pp 372–382. https://doi.org/10.1109/ICDCS.2017.123
60. Walia NK, Kaur N, Alowaidi M, Bhatia KS, Mishra S, Sharma NK, Sharma SK, Kaur H (2021) An energy-efficient hybrid scheduling algorithm for task scheduling in the cloud computing environments. IEEE Access 9:117325–117337
61. Alzahrani A, Tang M (2024) Energy-aware microservice-based saas deployment in cloud data center using hybrid particle swarm optimization. IEEE Access 12:45678–45692. https://doi.org/10.1109/ACCESS.2024.3462894
62. Mungoli N (2023) Scalable, distributed ai frameworks: Leveraging cloud computing for enhanced deep learning performance and efficiency. arXiv:2304.13738

Authors and Affiliations

Shahzeb Alam[1] · Muhammad Atif Ramzan[2] · Muhammad Zubair[3] · Ubaid Ullah[4] · Ziaullah[5] · Rab Nawaz[6] · Rahmat Ullah[6]

✉ Shahzeb Alam
salam@ebttikar.com

✉ Rahmat Ullah
rahmat.ullah@essex.ac.uk

Muhammad Atif Ramzan
atif.ramzan@composesolution.com

Muhammad Zubair
m.zubair3@lancaster.ac.uk

Ubaid Ullah
ukhan@deem.sa

Ziaullah
ziaullah@centraltrust.co.uk

Rab Nawaz
rab.nawaz@essex.ac.uk

1 Research and Development (R&D) Department, Ebtikar Technology Company Ltd, Riyadh 11573, Kingdom of Saudi Arabia

2 Department of Electrical Engineering, National University of Computer and Emerging Sciences, Islamabad 44000, Pakistan

3 School of Engineering, Lancaster University, Lancaster LA1 4YW, UK

4 Department of Telecommunications Engineering, University of Engineering & Technology Peshawar, Peshawar, KP 23200, Pakistan

5 Central Trust Limited, London WD24 4YW, England, UK

6 School of Computer Science and Electronic Engineering (CSEE), University of Essex, Colchester CO4 3SQ, England, UK