

Original Article

Reinforcement Learning-Based Resource Optimization for Autonomous Intelligent Computing Platforms

Sai Kishore

Independent Researcher, USA.

Received: 06-06-2026

Revised: 06-24-2026

Accepted: 07-02-2026

Published: 07-05-2026

ABSTRACT

To support diverse application workloads in dynamic environments with performance, energy efficiency, reliability and cost constraints, the autonomous intelligent computing platforms must be able to manage workloads that are heterogeneous. In the past, there were several challenges to overcome when using conventional scheduling methods, such as constantly changing workloads, varying hardware set-ups and different service-level targets. In this article, a reinforcement learning (RL) based resource optimization framework is proposed, which combines workload characterization, state estimation, adaptive allocation, energy-aware scheduling, and energy-driven policy improvement. Resource management is expressed as a Markov decision process with processor utilization, memory pressure, availability of accelerators, queue length, workload urgency, energy consumption as well as service-level indicators. A composite reward function is used to guide a deep reinforcement learning agent to decide resource allocation, task placement, container scaling, migration, and prioritization decisions. Constraint-aware action filtering and workload forecasting increases the safety and stability of the operations. Evaluation of the framework is conceptual; it includes the following metrics: throughput, latency, energy consumption, resource utilization, service-level violation and optimization overhead. Comparative numerical values are indicative, and need experimental confirmation. The proposed method lays the groundwork for resource management in intelligent cloud-edge infrastructures that is adaptive, scalable, and reliable.

KEYWORDS

Reinforcement Learning, Resource Optimization, Autonomous Computing, Deep Learning, Adaptive Scheduling, Energy Efficiency, Cloud-Edge Computing.

1. INTRODUCTION

1.1. Background

Applications like machine learning, autonomous systems, industrial automation, real-time analytics and digital twins are examples of applications that require a lot of computing resources, which can be provided by modern intelligent computing platforms. They are deployed in a variety of different infrastructures including CPU, GPU, memory, storage, cloud servers, and edge devices, where efficient resource coordination is a key requirement for performance, energy efficiency, and cost savings. Static allocation, rule-based scheduling, mathematical optimization, and threshold-based autoscaling are conventional resource management strategies that tend not to fit the needs of unpredictable workloads and competing application requirements. As a result, these techniques can lead to inefficient usage of resources, slower processing times, higher energy usage, and service level breaches during peak periods. Reinforcement learning (RL) is an adaptive method that allows an agent to learn resource allocation strategies through interaction with the environment and reward feedback. Deep reinforcement learning is another approach that incorporates neural networks with RL to handle the high-dimensional state of the system and a wide variety of allocation actions and thereby aids with complex decision making. Hence, incorporating RL into autonomous computing systems is a promising future approach to dynamically achieve optimal resource utilization, reduce operation cost, enhance application responsiveness, and guarantee reliable performance when workload conditions vary.

1.2. Problem Statement

An autonomous intelligent computing platform must be able to allocate heterogeneous resources dynamically in response to changing workloads while providing applications with varying hardware availability and priorities, as well as varying task execution times and loads, while the tasks are subject to uncertainty. The biggest difficulty is creating an intelligent controller that can optimize throughput, minimize latency and energy use while meeting service level requirements without requiring constant manual monitoring.

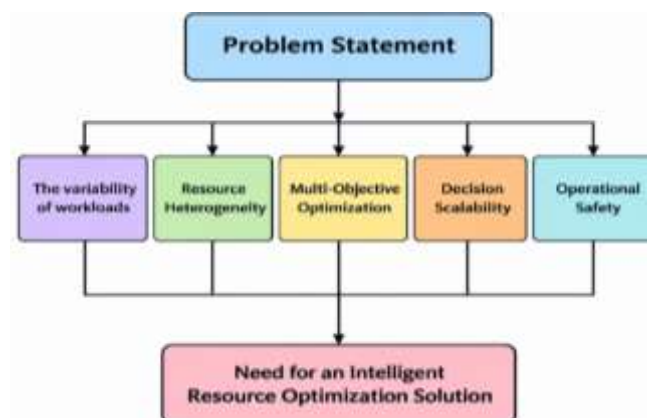


Fig 1 - Problem Statement and Key Challenges in Reinforcement Learning-Based Resource Optimization for Autonomous Intelligent Computing Platforms

1.2.1. The Variability of Workloads

The number of tasks, their processing needs and execution times vary depending on application needs and operating conditions. Such fluctuations may lead to certain resources being underutilized or unavailable, necessitating adaptive scheduling methods that can effectively navigate fluctuating workloads.

1.2.2. Resource Heterogeneity

The modern computing platform merges CPUs, GPUs, memory systems, cloud servers, and edge devices all with varying resources and processing power. To allocate resources efficiently, you need to choose the right resources for the job, take into account the compatibility of hardware and software, and consider communication needs.

1.2.3. Multi-Objective Optimization

There are competing goals for throughput and resource efficiency, while reducing latency, energy usage and cost. The controller needs to combine these performance metrics in a well-crafted reward function to account for the priorities of the applications and the services' level requirements.

1.2.4. Decision Scalability

In large computing environments, there are many possible allocation and scheduling actions and large dimensionality of the state space. The proposed controller should efficiently process system information and make quick decisions without adding too much computation overhead when the infrastructure grows.

1.2.5. Operational Safety

Resource allocation should be done with respect to the capacity of hardware resources, task priorities, application deadlines, and service level agreements. Hence there is a need for explicit feasibility checks and safety mechanisms in the framework to avoid invalid allocations and reliable system operation.

1.3. Research Objectives

The main goal of this research is to design a single reinforcement learning based resource optimization application for autonomous intelligent computing platforms which can dynamically manage the resources present in a platform while maintaining computational efficiency, energy sustainability and operational reliability. The first goal is to model this resource allocation problem as a sequential decision making problem by using a Markov Decision Process (MDP), which allows the controller to observe the current conditions of the system, choose the proper allocation action, and learn from the feedback on the performance of that action. The second goal is to create a deep reinforcement learning-based architecture that can adaptively schedule and control resources in various computing environments such as cloud data centers and edge computing nodes. This architecture seeks to meet the needs of fluctuating workloads, unpredictable resource allocation, and evolving application needs. The third goal is to develop a multi-objective reward function which

takes into account both throughput, latency and energy consumption, operational cost, use of resources and service level agreement compliance. For balancing competing performance requirements, appropriate reward weighting and normalization will be used based on the application priorities. The fourth goal is to embed constraint-aware action selection and safety enforcement mechanisms to avoid infeasible resource allocations, incompatibility of hardware and violation of predefined operational limits. The fifth goal is to develop a systematic comparison of the proposed framework with the existing resource allocation and threshold-based autoscaling and heuristic scheduling approaches under fixed workloads. Task completion latency, aggregate throughput, energy consumption per task completed, resource utilization, service-level violation rate, and controller overhead will be used to measure performance. The sixth objective is to understand the practical implementation requirements to deploy the proposed framework in heterogeneous telemetry collection, policy inference, computational overhead management, workload generalization and continuous performance monitoring in the cloud-edge environment. Overall, these goals are meant to create a scalable and flexible resource management approach that optimizes performance while adhering to operational requirements. Further, the research will help establish limitations and challenges in implementation by carrying out a systematic evaluation, which will serve as a basis for further investigation of reliable, energy-efficient, and autonomous management of computing resources.

2. LITERATURE SURVEY

2.1. Allocation and Scheduling of conventional Resources

In cloud computing, data centers, edge devices and distributed intelligent platforms, conventional resource allocation and scheduling techniques have been a critical part of computational resources management. Traditional scheduling algorithms such as First-Come-First-Served (FCFS), Shortest-Job-First (SJF), Round-Robin (RR) and priority based scheduling are based on a fixed set of rules and/or application requirements. While easy to use and have a known behavior, such methods can be difficult to scale for dynamic application requirements, diverse computing resources, and volatile workloads. Static resource allocation enables applications with fixed computational resources and can waste resources in periods of low demand, but on the other hand, fail to provide them in periods of high demand. Dynamic threshold-based techniques make them more adaptable by scaling resources based on CPU utilisation, memory usage, or network traffic; these techniques can result in frequent scaling operations, but they can also cause unnecessary scaling operations when they oscillate around predefined thresholds, resulting in higher operational costs. Metaheuristic optimization methods, like genetic algorithms or particle swarm optimization, can also be used to determine the best allocation strategies by testing several resource configurations, but can be too time-consuming for real-time decision-making. Machine learning techniques offer enhanced features, such as workload pattern recognition and forecasting resource needs based on past data. However, supervised learning techniques rely on good quality training sets and do not explicitly maximize system performance over the long-term. An alternative is reinforcement learning (RL) which learns allocation policies by continually interacting with the computing environment and making decisions based on the cumulative effects of the operation.

2.2. Reinforcement learning is the other type of AI applied in scheduling

Reinforcement learning is a promising technique for intelligent scheduling as it allows computing systems to learn intelligent resource management policies based on their experiences and feedback. For an RL problem, the agent is given a current state of a system, chooses an action, is given a reward (numeric), and then updates its policy to learn from the reward. While classical algorithms like Q-learning work well for small and discrete state-action space, deep reinforcement learning uses neural networks to tackle complex and high-dimensional computing environments. Mao et al. presented an RL-based scheduling system named Decima which learns scheduling policies based on the characteristics of workloads and job dependencies for data processing clusters [3]. As they show, learning-based methods can be used to enhance scheduling decisions in a setting where typical scheduling heuristics can fail in complex execution scenarios. The deep Q-networks (DQN) [4] are a hybrid of deep neural networks and Q-learning, used to estimate the action values and help decision making in larger state spaces. In policy-gradient methods, the parameterized policy is directly optimized, whereas Proximal Policy Optimization (PPO) enhances the stability of the training process by using a clipped objective that avoids overly large policy updates [5]. Actor-critic architectures are especially helpful in cases where resource allocation includes both discrete and continuous controls: workload placement, CPU shares, memory bounds, power control, etc. The performance of these algorithms however, relies on the environment, training, exploration strategies, and computational load. The choice of the right RL algorithm depends on the workload, resource limitations, and goals of the target platform.

2.3. Energy-Aware Cloud and Edge Resource Management

In an energy-aware resource management, the aim is to minimize the power consumption of the computational resources, while keeping the application performance, reliability of services and availability of the resource acceptable. For cloud data centers, energy usage is affected by server utilization, processor frequency, workload placement, cooling requirements and active number of computing nodes. Application consolidation can help save energy by running applications on fewer machines and putting unused machines into low-power mode. But too much consolidation can lead to greater contention for resources, processing delays, thermal stress, and potential service-level agreement violations. Edge computing raises more challenges due to the limited computing power, energy supply, and connectivity of edge nodes. Therefore, resource allocation needs to take into account not only computation costs but also the communication latency, available bandwidth, energy supply, and relative computation power of edge and cloud infrastructures. Reinforcement learning can assist these decisions by determining when to run workloads locally, offload workloads to cloud servers, and distribute applications across distributed nodes. The use of deep reinforcement learning in cloud resource scheduling has already been discussed in literature, and it is proven that it can be used to optimize the resource allocation scheme in the dynamic environment of a cloud service [1] [6]. However, policies must take into consideration changing workload distributions, scalability, inference overhead and energy consumption in order to be practically deployed. Therefore, an energy-aware controller should be designed based on a multi-objective optimization approach that

optimizes both energy efficiency and application performance, rather than just optimizing on energy efficiency.

2.4. Research Gap

Although substantial development has been made in resource management using RL, there still exist some research challenges in autonomous intelligent computing platform. Many of the existing solutions focus on individual performance measures such as energy consumption, job completion time or response latency, but fail to account for the conflicting demands of throughput, utilization, operational cost and service-level agreement (SLA) compliance. Secondly, an RL policy that is trained with limited workload may perform poorly in an environment with fluctuating demands, widely varying hardware, varying network conditions, or a new application. Third, unconstrained policy enforcement may result in unsafe policy enforcement decisions that result in an allocation that consumes more CPU, memory, storage, or power than is available, which may negatively impact system stability and application reliability. Fourth, the computational burden of policy inference, system monitoring and continuous model updating are not always fully accounted for in the performance assessment. In order to overcome these limitations, an RL-based resource optimization framework which integrates heterogeneous resource monitoring, multi-objective reward design, workload-adaptive decision making, and constraint-aware action filtering is proposed. The framework assesses the characteristics of the system based on system conditions like CPU utilization, memory availability, workload arrival rate, response latency, energy consumption and resource requirements, allowing the learning agent to choose appropriate allocation strategies. A separate safety enforcement system pre-validates, prior to execution, the proposed actions by the agent against the resources' capacity limitations and the operational requirements of the specific application. The separation of learned decision making and safety enforcement is designed to enhance adaptability and to avoid impossible resource assignments. The proposed direction is thus focussed on balance optimisation, operational reliability, and scalability in autonomous computing environments.

3. METHODOLOGY

3.1. Research Framework

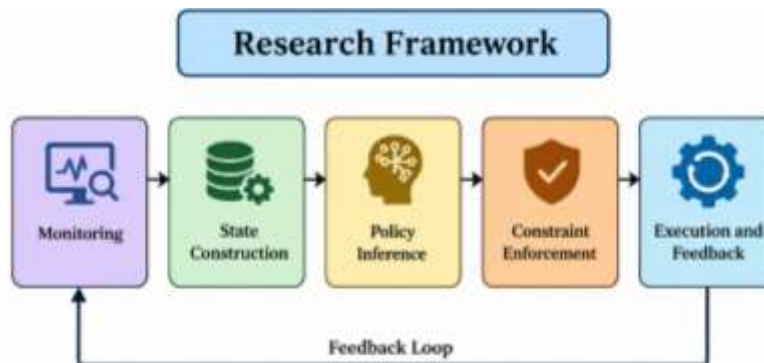


Fig 2 - Research Framework

3.1.1. Monitoring

The monitoring module gathers real-time telemetry on the computing nodes, applications, and network interfaces to detect resource availability and workload changes. It monitors CPU utilization, memory usage, processing delays, and energy usage and service level metrics. The gathered information gives a new picture of the computing environment. The accuracy of monitoring allows the RL controller to respond appropriately to the different operational conditions.

3.1.2. State Construction

The gathered telemetry is processed, normalised and converted to a state vector, structured for processing by the neural network. The state representation is a weighted mix of resources available, workload intensity, queue length, task priority and performance metrics. Normalization helps to minimize feature-scale differences and facilitates stable policy learning. The information generated in this stage will enable you to make informed decisions about resource allocation.

3.1.3. Policy Inference

The RL agent is able to analyze the current state and formulate candidate actions, including scaling resources, scheduling tasks, moving workloads, and adjusting allocations. It is designed to maximize cumulative rewards of performance, energy efficiency and resource utilization. The chosen action takes into account workload needs and expected system conditions. This allows for adaptive schedules in contrast to static schedules.

3.1.4. Constraint Enforcement

The constraint enforcement module ensures that every candidate action meets computing capacity, memory, power and application-specific service requirements. Actions that are impossible to perform are rejected or adjusted prior to execution in order to avoid overcommitting resources. The safety rule works outside of the learned policy for safe operations. This approach decreases the risk of resource conflicts and service level breaches.

3.1.5. Execution and Feedback

Once actions are approved, they are executed by the resource management interfaces of the platform, and then the impacts of these actions are assessed by new performance and energy metrics. The outcomes observed are translated to reward signals to inform later policy improvements. Continuous feedback allows the agent to adjust to various workload and resource conditions. Districting decision-making from enforcement of execution helps to facilitate scalable, reliable and efficient resource optimization.

3.2. Proposed Reinforcement Learning Architecture

The proposed reinforcement learning architecture aims to adaptively optimise the allocation and scheduling of resources in autonomous intelligent computing platforms, which would continuously adjust to the varying workloads and heterogeneous resource availability. The system is modeled as a Markov Decision Process (MDP):

$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$. $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $P(s_{t+1} | s_t, a_t)$ is the state transition probability function, $R(s_t, a_t)$ is the reward function, and γ is the discount factor. The controller receives observations of the current system state at each decision step t , chooses a proper resource allocation action, and gets feedback on the system state based on the performance of the action. The state vector is $s_t = [u_t^{\text{cpu}}, u_t^{\text{mem}}, u_t^{\text{gpu}}, q_t, \lambda_t, e_t, l_t, d_t]$, which includes the CPU, memory and GPU utilization, the queue length, workload arrival intensity, energy consumption, observed latency and deadline or service level pressure. These parameters can offer an all-encompassing picture of the computing environment and help the agent determine resource constraints, workload fluctuations and possible performance degradation. The action vector is defined as $a_t = [x_t^{\text{place}}, c_t^{\text{cpu}}, m_t^{\text{mem}}, g_t^{\text{gpu}}, p_t^{\text{prio}}]$, the task placement, CPU allocation, memory allocation, GPU assignment and task priority, respectively. These decisions can resort to action spaces, which can be discrete, continuous or hybrid depending on the computing platform. A learning mechanism called an actor-critic architecture is used to discover good resource management policies, where the actor proposes actions and the critic predicts the long-term total reward to be obtained for taking the action. It could be incorporated as a stability enhancement technique for training, known as the proximal policy optimization (PPO) algorithm [5]. The reward function can include throughput, latency, energy costs, use of resources and service level compliance to promote a balanced optimization. In addition, a constraint-aware action filtering mechanism checks the proposed decisions against the physical resource limits and the application requirements before being executed. The architecture's goal is to optimize resource usage and minimize operational expenses while ensuring consistent performance, all through the process of repeated interactions, reward evaluation, and policy updates. The mathematical model that is proposed solves the problem of resource allocation in autonomous intelligent computing platforms as an optimization problem using constrained reinforcement learning. The goal is to maximize the expected discounted cumulative reward over a finite horizon, which in this case is a predefined time T , and reward is denoted by r_t obtained at time step t and a discount factor γ is used and expressed as $\pi^* = \arg \max_{\pi} \mathbb{E}_{\pi} [\sum_{t=0}^{T-1} \gamma^t r_t]$. The reward function combines several performance metrics like throughput, latency, energy usage, operational cost, service level violations and resource usage. It is specified as: $r_t = w_1 \widehat{T}_t - w_2 \widehat{L}_t - w_3 \widehat{E}_t - w_4 \widehat{C}_t - w_5 V_t + w_6 U_t$, where the normalized variables are their respective performance measures and the w_i are the nonnegative weighting coefficients. These weights are set based on the application needs, enabling low latency application, high throughput for batch processing applications and energy efficiency. Normalization allows for minimizing the impact of measurement differences on policy learning. Physical resource constraints, such as CPU capacity, memory availability, accelerator compatibility, task dependencies, and network limitations must be met in the optimization process. The total of all allocations to task j for resource type i should not exceed the capacity of that resource, which is given as $\sum_{j=1}^N c_{ij}(t) \leq C_i$. Likewise, memory allocated should be within the memory limitation of the node. The constrained optimization formulation includes allowed resource usage and service-level violations. Proximal Policy Optimization (PPO) uses clipped objective that

helps prevent too big of steps, and uses the probability ratio and the estimated advantage function to make training more stable when learning a policy. The key point to note is that while reward optimization can ensure constraint satisfaction this is not sufficient, and the framework explicitly includes action validation and safety enforcement prior to the allocation of resources. The integrated mathematical formulation enables the balanced performance optimization, operational reliability and efficient use of resources under dynamic load conditions.

3.3. Optimization Algorithm and Autonomous Workflow

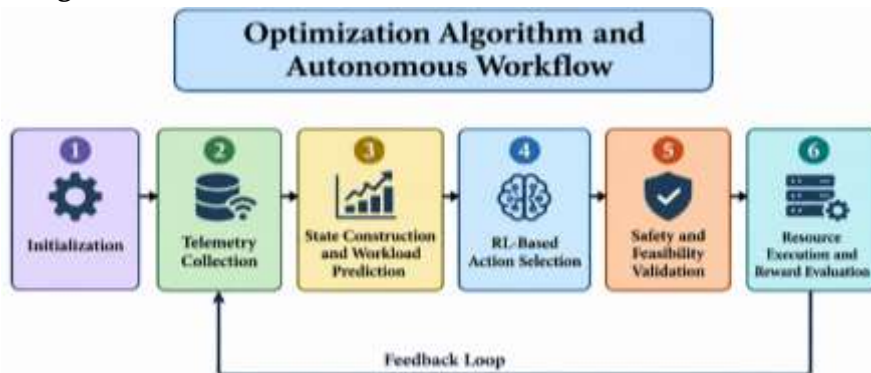


Fig 3 - Optimization Algorithm and Autonomous Workflow

3.3.1. Initialization

The computing platform, reinforcement learning policy and resource management environment are initialized. Available CPU, memory, GPU, workload limits and trained model parameters are set to provide initial operating conditions.

3.3.2. Telemetry Collection

The monitoring module gathers relevant data from the system such as CPU usage, memory usage, availability of GPUs, queue length, response latency, energy usage and service-level indicators. These measurements give up-to-the-minute data on resources available and performance of the applications.

3.3.3. State Construction and Workload Prediction

The telemetry gathered is processed and converted to a normalized state vector for the RL agent. Resource utilization, task arrival rates, queue conditions and application deadlines are used to estimate workload pressure to help decision making on proactive allocation.

3.3.4. RL-Based Action Selection

RL actor chooses appropriate resource management actions from the current state. These include workload placement, resource scaling, CPU and memory allocations, GPU assignment and workload scheduling for max performance and energy efficiency.

3.3.5. Safety and Feasibility Validation

The proposed actions are benchmarked to the capacities of the physical resources, to hardware compatibility, to priorities for the task, and to constraints on operations. If a decision is infeasible, then it is rejected or modified prior to execution to avoid service-level violations and resource overcommitment.

3.3.6. Resource Execution and Reward Evaluation

The validated allocation decisions are implemented; the resulting system performance is measured. The reward for assessing the effectiveness of each action is calculated as throughput, latency, energy consumption, resource utilization and service-level compliance.

4. RESULT AND DISCUSSION

4.1. The experimental setup

A controlled cloud-edge computing environment that includes a combination of computing nodes, workload patterns and applications with diverse performance needs should be used to evaluate the proposed reinforcement learning based resource optimization framework. The experimental setting should encompass typical conditions such as batch processing, latency-sensitive services, and computationally demanding AI workloads. A workload trace needs to contain a mix of steady state workload, varying resource needs, and sudden bursts of new task arrivals to evaluate the adaptability and robustness of the proposed approach. The constraint-aware deep reinforcement learning approach is contrasted with four existing resource management strategies: static resource allocation, threshold-based autoscaling, heuristic workload-aware scheduling, and the proposed constraint-aware deep reinforcement learning approach. All strategies should be executed with the same workload traces, the same number of resources, the same priorities, and the same service-level agreement (SLA) requirements to make the comparison fair. Multiple runs of each experiment using multiple random seeds should be performed to assess variability in workload execution and model training and the performance measurements analyzed accordingly. Evaluation metrics include average task completion latency, aggregate throughput, energy consumption per completed task, resource utilization, SLA violation rate and controller overhead. Throughput is the number of tasks successfully completed per unit time and latency is the time to complete the tasks. Energy consumption per completed task measures computational efficiency, and resource usage measures the effectiveness of use of the available infrastructure. SLA violation rate is the percentage of tasks that do not meet the required service level, and controller overhead is the processing overhead for controller monitoring and decision making. The amount of improvement should be measured against each baseline with the same measurement intervals. If the metric is lower-the-better, percentage improvement is calculated as $(I_M = \frac{M_b - M_p}{M_b} \times 100\%)$, where (M_b) is the baseline measurement and (M_p) is the proposed method measurement. If the metric is higher-is-better (e.g., throughput) then improvement is defined as $(I_T = 100\% \times \left(\frac{T_p - T_b}{T_b}\right))$. Last but not least, mean values, confidence intervals and the relevant statistical significance tests should be reported to see if the performance differences are reliable. The performance improvement is a number and the results will vary depending on the implementation

and workload settings, so that the results of the numbers must be shown after experiments or validated simulations are done.

4.2. Comparative Percentage Analysis

The following table presents an illustrative percentage-based comparison, not measurements obtained from an implemented experiment. The figures are hypothetical examples intended to demonstrate the reporting format and interpretation of a potential evaluation. They must be replaced with measured results before the manuscript is submitted as an empirical research article.

Table 1: Illustrative Relative Improvement over Static Resource Allocation

Performance Metric	Threshold-Based Scheduling	Heuristic Scheduling	Proposed Constrained DRL
Throughput improvement	8%	13%	20%
Latency reduction	10%	16%	24%
Energy consumption reduction	5%	11%	18%
Resource utilization improvement	7%	12%	17%
SLA violation reduction	12%	19%	28%

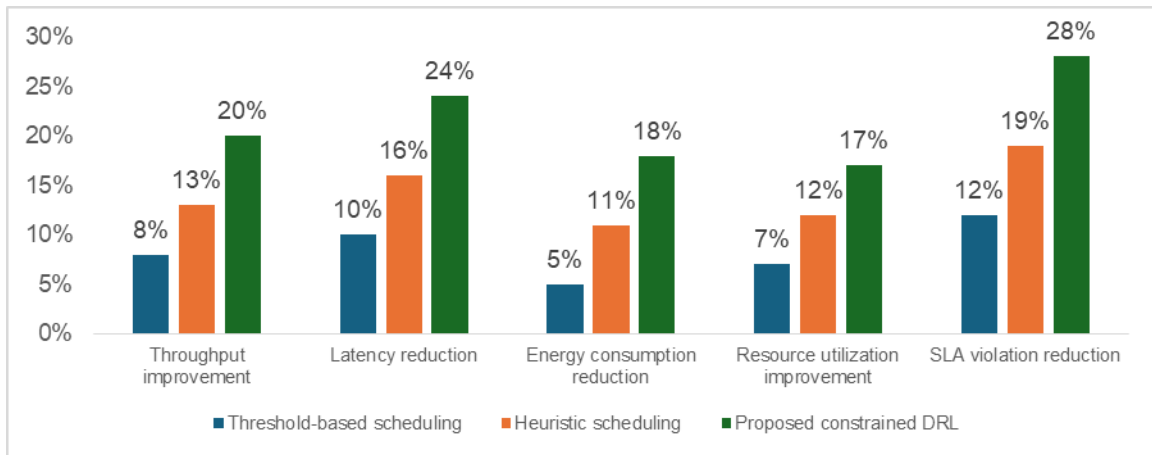


Fig 4 - Illustrative Relative Improvement over Static Resource Allocation

4.3. Discussion of Expected Performance and Limitations

The proposed architecture is designed to improve decision quality by integrating workload observations, long-term reward optimization, and feasibility checks. Its principal expected advantage over static scheduling is the ability to adapt allocations when resource demand changes. Compared with threshold-based autoscaling, a learned policy can potentially account for interactions between multiple resource types and the delayed consequences of allocation decisions. However, these benefits are conditional rather than guaranteed. A poorly trained agent may overallocate resources, oscillate between configurations, or optimize the reward function at the expense of an inadequately represented objective. Reward normalization, representative workload training, and systematic

hyperparameter selection are therefore essential. Constraint enforcement is equally important. A policy that produces high cumulative reward may still be unsuitable for production if it violates hard latency, capacity, or availability requirements. The feasibility layer must be evaluated independently by measuring invalid-action frequency, fallback activations, and constraint violations. The controller itself introduces inference, monitoring, and training overhead. Consequently, net performance should include the resources consumed by the optimization system. A meaningful evaluation must also test generalization to unseen workloads, bursty arrivals, node failures, and changing hardware availability. Overall, the proposed framework establishes an implementable research direction for autonomous resource management. Its practical value must be determined through reproducible experiments rather than inferred solely from the mathematical formulation or illustrative comparison.

5. CONCLUSION

The research proposed a reinforcement learning framework for resource optimization in autonomous intelligent computing platform that tackles the problem of dynamic workloads, heterogeneous computing resources, conflicting performance requirements and operation constraints. The proposed architecture equips the system with various capabilities for intelligent and automated resource management, such as real-time resource monitoring, deep reinforcement learning, adaptive scheduling, and constraint-aware action execution. The resource allocation problem was represented as a Markov Decision Process (MDP) and the controller was allowed to observe the state of the system, choose an appropriate allocation action, and learn from the performance feedback. The framework is based on an actor-critic learning architecture and a Proximal Policy Optimization (PPO)-based objective, which enable adaptive decision-making in complex computing environments. A multi-objective reward function was designed that takes into account throughput, latency, energy consumption, operational cost, resource utilization, and service-level agreement (SLA) compliance. In addition, explicit capacity constraints and feasibility validation mechanisms were added to minimise the possibility of being able to assign an invalid resource and enhance operational reliability. The methodology also paved a systematic way for evaluating and comparing the proposed approach to static resource allocation, threshold-based autoscaling, and heuristic workload-aware scheduling. Average task completion latency, aggregate throughput, energy consumption for each completed task, resource utilisation, SLA violation rate, and controller overhead are taken into account for performance assessment under consistent workloads. The design of the evaluation also focuses on repeat experiments, statistical analysis, and experiments conducted at different workload intensities to ensure reliable comparisons. While the proposed architecture is a structured basis for adaptive resource optimization, its efficacy needs to be proven by implementation and repeatable experiments. Therefore, the illustrative percentage comparisons provided in this research should be considered as hypothetical and not experimental results. Future scope will include deployment in realistic cloud-edge scenarios, generalizing the policy with unseen workloads, and assessing the scalability for various heterogeneous computing nodes. Other research will focus on multi-agent reinforcement learning, safe policy adaptation, fault-tolerant resource allocation and energy-aware workload migration. Overall, the proposed framework provides a

methodology for the design of intelligent, scalable, and constraint-aware resource management systems which aim at achieving a balance between the computational performance, energy efficiency, operational cost, and service reliability of autonomous computing platforms.

REFERENCE

- [1] G. Zhou, W. Tian, R. Buyya, R. Xue, and L. Song, "Deep reinforcement learning-based methods for resource scheduling in cloud computing: A review and future directions," *Artificial Intelligence Review*, vol. 57, Art. no. 124, 2024, doi: 10.1007/s10462-024-10756-9.
- [2] Nangi, P. R. (2022). Multi-Cloud Resource Stability Forecasting Using Temporal Fusion Transformers. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, 3(3), 123-135.
- [3] Syed, Y. (2024). DevSecOps and Enterprise-Scale CI/CD Pipeline Engineering: Governing Delivery Velocity, Embedded Security, and Release Automation Across Polyglot Environments. *International Journal of Scientific Research in Science, Engineering and Technology*, 11(5), 475-484. DOI: <https://doi.org/10.32628/IJSRSET2513156>
- [4] Virendra Jangid. (2025). *AI-Powered Cross-Border Securities Processing and ADR Platform Optimization*. *International Journal of Leading Research Publication*, 6(11). <https://doi.org/10.70528/IJLRP.v6.i11.2233>
- [5] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proc. 15th ACM Workshop on Hot Topics in Networks (HotNets-XV)*, 2016, pp. 50-56.
- [6] Nangi, P. R., & Obannagari, R. N. CK, & Settipi, S.(2022). Predictive SQL Query Tuning Using Sequence Modeling of Query Plans for Performance Optimization. *International Journal of AI, BigData, Computational and Management Studies*, 3(2), 104-113.
- [7] Vasantha Kumar Hosahalli Seenappa. (2024). NG-iRTS: A Python and Ansible-Based Infrastructure as Code Framework for Enterprise Network Automation. *International Journal of Intelligent Systems and Applications in Engineering*, 12(23s), 4486-4492. Retrieved from <https://ijisae.org/index.php/IJISAE/article/view/8533>
- [8] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529-533, 2015, doi: 10.1038/nature14236.
- [9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017, doi: 10.48550/arXiv.1707.06347.
- [10] Nangi, P. R., Obannagari, C. K. R. N., & Settipi, S. (2022). Enhanced Serverless Micro-Reactivity Model for High-Velocity Event Streams within Scalable Cloud-Native Architectures. *International Journal of Emerging Research in Engineering and Technology*, 3(3), 127-135.
- [11] Tayal, C. (2026, January). Assessing the Scope of Optimising Operational Roles Through Intelligent Automation Frameworks. In *2026 IEEE 16th annual computing and communication workshop and conference (ccwc)* (pp. 0063-0068). IEEE
- [12] Syed, Y. (2025). Generative AI Integration and RAG-Based Enterprise Tooling: Architectures, Techniques, and Organizational Impact. *International Journal of Scientific Research in Science, Engineering and Technology*, 12(6), 600-606. DOI: <https://doi.org/10.32628/IJSRSET2613255>
- [13] Fung, W., Guo, L., Hilliard, D., Casadei, A., Ratan, R., Bhaduri, S., ... & Rakshit, A. (2025). OpComm: A Reinforcement Learning Framework for Adaptive Buffer Control in Warehouse Volume Forecasting. *arXiv preprint arXiv:2512.19738*.
- [14] H. Mao, S. B. Venkatakrishnan, M. Schwarzkopf, and M. Alizadeh, "Decima: Learning scheduling algorithms for data processing clusters," in *Proc. ACM SIGCOMM Conf.*, 2019, pp. 634-646.
- [15] Nangi, P. R., & Settipi, S. (2023). A cloud-native serverless architecture for event-driven, low-latency, and AI-enabled distributed systems. *International Journal of Emerging Research in Engineering and Technology*, 4(4), 128-136.
- [16] Visweswaran, K. (2026, March). Low-Latency AI Models for Predictive Connection Management in Bluetooth Low Energy (BLE) Communication Systems. In *2026 6th International Conference on Expert Clouds and Applications (ICOECA)* (pp. 541-548). IEEE. <https://doi.org/10.1109/ICOECA68095.2026.11485045>
- [17] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397-1420, 2012, doi: 10.1002/cpe.1867.
- [18] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322-2358, 2017.

- [19] Pavan Nithin Mullapudi (2024). An Adaptive Explainability Framework for Machine Learning Predictions of Deals in Cloud Computing . International Journal of Multidisciplinary Research and Growth Evaluation (IJMRGE), 5(3), 1027-1034. DOI: <https://doi.org/10.54660/IJMRGE.2024.5.3.1027-1034>
- [20] Reddy Nangi, P., & Reddy Nala Obannagari, C. K. (2023). Scalable End-to-End Encryption Management Using Quantum-Resistant Cryptographic Protocols for Cloud-Native Microservices Ecosystems. *International Journal of Emerging Trends in Computer Science and Information Technology*, 4(1), 142-153.
- [21] Mullapudi, P. N., Bhaduri, S., Casadei, A., Guo, L., Wang, H., Ashok, V., ... & Rakshit, A. (2025). An end-to-end Causal Modeling Framework for Advanced Attribution in Supply Chain Operations. In *KDD 2025 Workshop on AI for Supply Chain: Today and Future*.
- [22] X. Zhou, J. Yang, Y. Li, S. Li, and Z. Su, "Deep reinforcement learning-based resource scheduling for energy optimization and load balancing in SDN-driven edge computing," *Computer Communications*, vols. 226-227, Art. no. 107925, 2024, doi: 10.1016/j.comcom.2024.107925.
- [23] Casadei, A., Bhaduri, S., Malshe, R., Mullapudi, P., Ratan, R., Pole, A., & Rakshit, A. (2025). Scaling Causal Mediation for Complex Systems: A Framework for Root Cause Analysis. *arXiv preprint arXiv:2512.14764*.
- [24] Y. Syed, "Enterprise Observability and Software Delivery Performance: From Telemetry to Engineering Intelligence", *IJADSMC*, vol. 5, no. 2, pp. 01-08, Sep. 2022, doi: 10.67228/30713498/IJADSMC-2022PIIKP2M
- [25] H. Hu, V. Chang, and M. Chen, "Deep reinforcement learning for performance-aware adaptive resource allocation in mobile edge computing," *Wireless Communications and Mobile Computing*, 2020, Art. no. 2765491, doi: 10.1155/2020/2765491.
- [26] Nangi, P. R., Obannagari, C. K. R. N., & Settipi, S. (2022). Self-auditing deep learning pipelines for automated compliance validation with explainability, traceability, and regulatory assurance. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, 3(1), 133-142.
- [27] M. Wen, X. Liu, X. Ning, C. Liu, X. Chen, J. Nian, and L. Cheng, "Deep reinforcement learning for energy-efficient workflow scheduling in edge computing," *Computer Networks*, vol. 274, Art. no. 111790, 2026, doi: 10.1016/j.comnet.2025.111790.
- [28] Nangi, P. R., & Obannagari, R. N. CK, & Settipi, S.(2024). A Federated Zero-Trust Security Framework for Multi-Cloud Environments Using Predictive Analytics and AI-Driven Access Control Models. *International Journal of Emerging Research in Engineering and Technology*, 5(2), 95-107.
- [29] Syed, Y. (2023). Polyglot Persistence and Enterprise Data Migration: Architectures, Consistency, and Practice. *American International Journal of Computer Science and Technology*, 5(3), 61-66. <https://doi.org/10.63282/3117-5481/AIJCST-V5I3P106>
- [30] "Federated deep reinforcement learning for energy-efficient edge computing offloading and resource allocation in Industrial Internet," *Applied Sciences*, vol. 13, no. 11, Art. no. 6708, 2023, doi: 10.3390/app13116708.
- [31] Nangi, P. R., Obannagari, C. K. R. N., & Settipi, S. (2024). Serverless computing optimization strategies using ML-based auto-scaling and event-stream intelligence for low-latency enterprise workloads. *International Journal of Emerging Trends in Computer Science and Information Technology*, 5(3), 131-142.