National College of Ireland

Multi-Agent Reinforcement Learning based Predictive Scheduling and Resource Allocation in AWS Edge-Cloud Kubernetes Environments

MSc Research Project
Cloud Computing

Radha Kantipudi
Student ID: 23226391

National College of Ireland

Supervisor: Shaguna Gupta

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Radha Kantipudi
Student ID:	x23226391
Programme:	Dept. of MSc Cloud Computing
Year:	2024 to 2025
Module:	MSc Research Project
Supervisor:	Shaguna Gupta
Submission Due Date:	01/09/2025
Project Title:	Multi-Agent Reinforcement Learning based Predictive Scheduling and Resource Allocation in AWS Edge-Cloud Kubernetes Environments
Word Count:	2989
Page Count:	27

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Radha Kantipudi
Date:	31th August 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Multi-Agent Reinforcement Learning based Predictive Scheduling and Resource Allocation in AWS Edge-Cloud Kubernetes Environments

Radha Kantipudi
x23226391

Abstract

The optimization of task scheduling and placement in Kubernetes based containerized cluster environments presents serious difficulties imposed by the heterogeneous constraint on resources, the dynamic nature of workloads, and compounding requirements on performance goals. Although the performance of the existing single-agent deep reinforcement learning (DRL) methods such as Proximal Policy Optimization with Least Response Time (PPO-LRT) on the issue of load balancing has improved, it does not satisfactorily consider a distributed decision-making environment, nor does it support the multi-objective optimization requirements of modern containerized systems. This research presents a multi- agent deep reinforcement learning (MADRL) system that extends to the existing traditional single-agent systems to add additional specific agents with the purpose of optimising response time, balancing load and privacy protection. The framework incorporates LSTM-based predictive scaling to undertake proactive resource management and performs intelligent node selection decisions depending on workload characteristics. The synthetic workload evaluation using PolybenchC-inspired benchmarks demonstrates that the experimental performance is better than the baseline Kubernetes scheduler and current PPO-LRT techniques, in terms of providing average response-time competitiveness, increased resource utilization efficiency (achieving 100% node utilization vs 75% for baseline), and better load balancing across cluster nodes while without violating the privacy requirements of sensitive task placement.

Keywords: deep reinforcement learning, edge-cloud computing, predictive resource management, scheduling, Kubernetes

1 Introduction

1.1 Research Background and Motivation

Container orchestration has lately become the essential framework for decentralized application management, allowing for managed resource usage and scaling applications in modern computing environments. The de-facto container orchestration platform, Kubernetes, is finding use in the automation of applications running across diverse cluster configurations. But the default Kubernetes scheduler cannot perform optimal resource allocation when there is heterogeneous workload demands and multi-objective optimization requirements (Bohm and Wirtz, 2022). The advent of deep reinforcement learning

(DRL) in recent years has already displayed encouraging performance in solving challenging tasks to schedule; single-agent frameworks (such as Proximal Policy Optimization) of DRL have already displayed significant gains in task scheduling performance (Wang et al., 2023). However, the fact that modern containerized environments require simultaneous optimization of multiple competing objectives implies that conventional single-agent algorithms might be incapable of managing and optimizing multiple competing goals simultaneously and dynamically adjust based on workload patterns and resource limitations found in diverse application scenarios (Kumar et al., 2021) which forms the basis of motivation for this research work.

1.2 Problem Statement

The existing task scheduling strategies in Kubernetes clusters are limited in a number of critical ways. The single-agent based reinforcement learning framework could work in simple cases of load balancing, but it would be insufficient and lack required flexibility to support the distributed decision structure necessitated by multi-objective optimization in containerized systems (Jayanetti et al., 2024). The present solutions to such challenges do not achieve optimal solutions to conflicting problems, including minimization of response time, resource efficiency, load balancing, and privacy preservation, leading to poor performance of diverse workloads and sub-optimal resource utilization (Shen et al., 2023). In addition, predictive resource management capabilities are not available, and this leads to reactive instead of proactive scaling decision, resulting in performance bottlenecks during workload surges commonly observed in dynamic containerized environments.

1.3 Research Question

In what ways, the proposed multi-agent deep reinforcement learning framework (MADRL) with predictive resource management in Kind-based Kubernetes clusters, can optimize task scheduling and resource assignment to obtain the better response time, load balance, and resource efficiency?

1.4 Problem Solution

This study presents an MADRL framework as a promising alternative to the single-agent method PPO-LRT that are limited by the complexity of single-agent solutions. In contrast to the single-agent PPO-LRT algorithm proposed by (Wang et al., 2023) whose priority is, in general, to balance the load and improve the response time, we use three collaborating agents: the response time agent for latency optimization, load balancing agent for resource allocation, and privacy agent for safe placement of tasks. It combines LSTM-based predictive scaling to proactively manage the resources and deploys the mechanisms of intelligent node selection, thus achieving the optimization qualities of multiple objectives and being more efficient in resource utilization than the original PPO-LRT algorithm. The framework is also executed and tested on Kind clusters installed on EC2 environments, which offers a containerized test environment to allow controlled experimentation and evaluation of performance.

1.5 Research Objective and Contributions

The main research objective is to design and test the MADRL framework that can work significantly better than the current single-agent solutions for task scheduling optimization in containerized Kubernetes systems so as to meet the multiple conflicting goals. The key contributions are as presented below:

• An MADRL framework capable of dynamically allocating specialized response-time, load-balancing and privacy optimizing agents to cluster nodes to allow distributed decision-making across heterogeneous container workloads.

• The combination of LSTM-based workload prediction with proactive scaling of resources for preventative maintenance of performance bottlenecks even before they actually would occur.

• An intelligent scheduling mechanism which dynamically finds an optimal assignment of a task across cluster nodes depending on workload properties and system limits to make optimal placement decisions.

• The development of a multi-objective optimisation engine to help balance between competing objectives in this context as well as accommodate to dynamic synthetic and diverse demands.

• A controlled experimental validation using PolybenchC-inspired synthetic workloads on Kind clusters to verify the improvements compared to the other current approaches and baseline research work in various performance dimensions.

1.6 Challenges and Limitations

There may arise many technical and practical bottlenecks with the proposed framework. The algorithmic problems may become prominent as the interaction of different agents must be synchronized, convergent, and stable in the containerized cluster environments (Lim and Joe, 2023). The problems concerning the scaling are associated with the application of the framework onto the distributed Kubernetes cluster; the deployment of which is described by the varied capacity to process computationally and network environment characteristics. The computational overhead due to the utilization of predictive components may impact on real-time decision-making among resource constrained cluster nodes. In this implementation, the multi-agent coordination mechanism introduces a 7.4% response time overhead in comparison to baseline scheduler due to the coordination mechanisms between the three agents, scheduling constraints imposed by the added privacy features, and computational complexity. With the increase in cluster size and the scalability challenges associated with that, the O(n) complexity also becomes problematic. The privacy preservation mechanisms employed have the potential to hinder the performance optimization requirements, in which there must be a thin line set between security and efficiency. Moreover, LSTM-based prediction relies on the quality and availability of historical workload data in the framework in the first place, and new cluster deployments might not have enough historical data.

1.7 Thesis Structure

The thesis, structured in six chapters, presents the evolution and evaluation of a MADRL framework for optimizing scheduling in Kubernetes. Chapter 2 reviews existing DRL systems in container orchestration and identifies the research gap in multi-objective optimization, while Chapter 3 outlines the methodology with Kind cluster configuration and

workload generation. Chapter 4 describes the MADRL design with three expert agents, an LSTM predictor, and an optimization engine, followed by Chapter 5 on execution using containerized microservices deployed in Kind clusters on EC2. Finally, Chapter 6 evaluates performance, showing that MADRL outperforms the baseline Kubernetes scheduler with improved resource utilization and privacy-aware scheduling while maintaining comparable response times and task completion rates.

2 Related Work

2.1 DRL based Task Scheduling in Edge-Cloud Environments

The work by (Zheng et al., 2022) presents a Deep Q-Network (DQN)-based workload-scheduling model applicable to edge computing systems and aimed at addressing the main task of workload balancing on the one hand, while reducing service durations and failed-task percentages on the other hand. The proposed approach was conducted in an environment consisting of dynamic edge devices with high levels of user mobility, and time-varying traffic patterns. The scheduling activity is formulated as a Markov Decision Process (MDP) in which the state spaces are defined in relation to the aspects of server usage, network specific conditions, and task features. The empirical findings from EdgeCloudSim reveal that DQN model demonstrates better results compared to benchmarking models, including PPO and DDPG because it cuts down average service times by 31 % when tested on a cluster of about 2000 mobile devices. However, the single-agent design of the DQN framework does not scale well as it cannot capture the decentralized decision-making process imposed by heterogeneous edge-cloud cluster, at least when many, competing goals must be acted upon simultaneously.

Wang et al., (2024) propose a Directed Acyclic Graphs (DAG) based application cost model to improve the load balancing and response time based on DRLs that can efficiently solve weighted cost optimization problems on an adaptive basis. They provide a feasible scheduler implementation to the FogBus2 serverless framework convenient to schedule IoT requests. The solution to their approach manages the complexity of interdependent task completion within the fog computing environment by performing better resource utilization than conventional scheduling algorithms. By conducting experiments on real test beds with actual IoT applications, a notable improvement in load balancing and response times was achieved. Nevertheless, the framework can be mainly considered a single-agent framework and does not focus on the multi-objective optimization needs of the emerging edge-cloud infrastructure, especially the privacy preservation and energy efficiency aspects, which are important for the industrial IoT implementations.

As presented by (Seid et al., 2021), the task offloading and resource allocation in multi-UAV enabled IoT edge networks are discussed through a multi-agent deep reinforcement learning (MADRL) methodology. Their work poses the problem as an extension of the MDP on stochastic games to achieve minimum long-term cost of computation in terms of energy cost and delay. The MADRL framework considers stochastic time-varying UAV channel strength and dynamic resource request to compute optimal resource allocation policies and computation offloading in aerial-to-ground network infrastructure. The experimental outcomes show that their proposed MADRL approach saves 38% and 55% average expenses over single-agent DRL and heuristic schemes, respectfully. Their multi-agent solution is capable of modelling the distributed decision-making process in edge environments that is not possible in single-agent approaches. Nonetheless, this frame-

work is limited to UAV-enabled networks and cannot directly be mapped to the more traditional edge-cloud Kubernetes deployment.

To overcome the shortcomings of its single-agent counterpart in situations that involve heterogeneous environments, (Kesavan et al., 2025) suggest the implementation of an MADRL framework as an optimization strategy to optimize task scheduling in Edge Kubernetes clusters. They use a three-agent system which includes response time agent, load balancing agent, and privacy agent, which is coordinated by an AI optimization engine that adapts competing goals dynamically. The architecture also combines the workload characterization, edge-cloud offloading policies and prediction-based scaling based on LSTM measures anticipating future workload requirements. Their experimental findings indicate better time concentration, equitable loads distribution and high success ratio tasks over conventional scheduling algorithms. The DQN-based method is, however, computationally costly because of the frequent updates and can be scalable in a large IoT setup with dynamic workloads.

Lilhore et al., (2025) propose a hybrid deep learning framework based on DQN and PPO to optimise IoT resources in a scalable manner for edge clouds. The hybrid system utilizes Graph Neural Networks (GNNs) to enhance the accuracy of the resources representations to provide reinforcement learning-based scheduling for better adaptation to changing workloads. Their method finds the best balance between policy learning and optimization, making their method to be used in modern low-latency high demand IoT systems like autonomous systems and real-time monitoring. However, the study mainly works on optimization of resources, whereas it does not strictly deal with issues pertaining to orchestration of containers in Kubernetes, and also have not been fully analyzed considering the multi-agent problem.

2.2 Multi-Objective Optimization in Multi-Agent Systems

Wang et al., (2025) propose a multi-access edge computing system, where a task offloading based on the MADRL is considered and handles the issue of finding the solution to the simultaneous task offloading and resource allocation optimization in the dynamic wireless environment where the resource capacity is limited at the edge. Their solution addresses a scenario of multi-user and multi MEC servers having diverse task demands and random channel conditions that solve offloading decisions, offloading ratios, and computing resources by focusing on minimizing the total energy consumption and time delay. The simulation results indicate 7-20% improvement over existing methods, but the framework majorly addresses conventional MEC conditions as opposed to container orchestration in the Kubernetes-based edge-cloud frameworks.

Zakaryia et al., (2025) suggest a collaborative optimization model that takes into consideration Distance to Task Location and Capacity (DTLC) in offloading tasks in multi-UAV assisted MEC scenarios with Multi-agent Deep Reinforcement Learning (MADRL). The framework involves advanced mechanisms of task offloading that takes into account not only the proximity but also the computational capacity in scheduling tasks. They address this problem using a multi-agent system that allows UAVs to coordinate to optimize resources allocation keeping the network communication efficient. Experimental evidence proves the higher effectiveness in decreasing service delays and increasing resource utilization in comparison with traditional centralized strategies. The research paper effectively deals with the complication of mobile edge corridors, which have dynamic topology alterations.

Danino et al., (2023) present a container allocation framework in cloud scenarios through MADRL to solve the problem of resources distribution associated with containerized applications. Using a decentralized allocation decision, the framework instates a multi-agent environment to manage an elevated complexity of action spaces, where each agent determines which container to put in and takes account of available resources such as CPU, memory and communication bandwidth. They come up with a collaborative process in designing states and reward to lead to efficient container assignment, studying both LSTM and memory-augmented agents to address the problem of container assignment. Based on experimental evidence, the technique promises up to 28 percent in improving execution time over available bin-packing heuristics and industry tools available using Kubernetes in cloud-based solutions and application, however, the performance when applied to edge-cloud hybrid environments with predictive scaling capabilities has not been tested.

Cicco et al., (2025) consider the problem of service orchestration in edge-cloud continuum setting where multi-objective reinforcement learning (MORL) is used to accommodate conflicting goals such the probability of accepting a service, the quality-of-service (QoS) offered, and the network energy consumed. Their work addresses the problem of coordination of services across the diverse landscapes, which scale between cloud data centers and the edge devices, and they understand well that the classic single-objective reinforcement learning frameworks are not sufficient to resolve the conflicts between various operational needs. The framework uses a set-based multi-objective reinforcement learning policy which does not need retraining to adapt to any network topology and includes energy consumption modeling of heterogeneous edge devices and servers over a number of possible workloads. The numerical evidence suggests that their MORL policy improves over baseline heuristics by 30 percent on average across wide-scopes of objective preferences and scales to networks with up to 5x as many nodes as training distributions.

Safavifar et al., (2024) introduce DEWOrch, a MADRL model that aims at high-efficiency workload orchestration in extreme edge computing settings where only resource-limited edge devices can be applied due to a lack of high-capacity MEC servers. The framework addresses workload orchestration over heterogeneous edge devices with a high degree of mobility and geographical dispersion, which have to operate within an environment with autonomy in terms of user request and a service level agreement. Experimental findings in smart manufacturing contexts show DEWOrch outperforms state-of-the-art with an approximate 50% reduction in resource waste and accompanying higher task success rates and reduced power consumption per task even though the methodology is targeted at extreme-edge context without a traditional MEC server network.

2.3 Predictive Resource Management in Containerized Edge-Cloud

Yang and Esquivel, (2024) suggest an adaptive LSTM network-based implementation of dynamic integration of intelligent end-edge-cloud systems to emphasize the Quality of Experience (QoE) by the use of an event-driven approach to automatically adjust the resource demands to the capacity of edge devices. In their study, they handle the inherent problem of limited computing power and storage at the edge device that cannot stream and query data in real-time and process them. The framework employs LSTM networks to examine edge device storage capacity in real-time and minimize uncertainty and partial adaptation of edge devices to the demands of the users. Experimental outcomes denote

that system responsiveness and resource utilization efficiency are highly enhanced with respect to conventional static allocation approaches. But their solution is more targeted at adapting storage capacity than at full task scheduling and does not directly consider the needs of container orchestration in the Kubernetes context.

Zhang et al., (2024) develop a MADRL-based optimization framework of edge-cloud collaborative computing resources management in Internet of Vehicles (IoV) scenarios. Their paper proposes to work on the problem of performance-scalable computational resource allocation in distributed edge-cloud infrastructure to enable the use of vehicular applications over such a communication network with low latency and high reliability. The framework uses multi-agent DRL methods to decide how resources can be coordinated between multiple edge servers by considering the dynamic nature of vehicular networks based on high mobility patterns. The empirical findings indicate better use of the resources and less service latency than conventional ways of allocating resources statically, and the framework is customized to the vehicular scenarios and might potentially need modifications as it is applied to the wider edge-cloud contexts beyond transport.

Marques et al., (2024) introduce pro-active resource management principles in Kubernetes based cloud systems that are more secured and results in more predictable scalability. Their work addresses the inadequacies of built-in Kubernetes monitoring tools by creating an end-to-end monitoring framework that is able to gather real-time custom performance data of microservices to circumvent standalone Kubernetes monitoring shortcomings. The framework integrates a load prediction service which can be used to perform proactive scaling operations, which anticipate future user session values and post as service metrics gathered by the monitoring system to provide predictive scaling activities. The findings indicate that the duration of threshold breaches can be decreased, and the responsiveness can be enhanced due to the proactive resource assigning.

To solve the data constraint and poor prediction of CPU utilization in edge decision engines, (Golec et al., 2025) add to the field, a generative AI-driven proactive Kubernetes container orchestration framework (GAIKube) to operate in heterogeneous edge environments. In their work, they address the issue of containerized edge computing where decision engines commonly use weak resource predictors with limited training data, which results in addressing delays, accuracy, and service-level agreements (SLA). The framework adopts a triple strategy: time-series CPU usage datasets are augmented with DoppelGANger (DGAN) to enable computationally diverse edge clusters, Google TimesFM is used to support long-horizon predictions for fast and accurate performance measures, and a dynamic container orchestrator is used to make scheduling, migration, and vertical scaling decisions. The GAIKube orchestrator considers conflicting goals of contrasting SLA violations, cost, and accuracy in making proactive decisions to prevent server faults based on the estimated resource use patterns. The experiments conducted on Google Kubernetes Engine (GKE) exhibit less than 4% percent SLA failure and about 4 percent accuracy-loss experienced by users. But the framework has significant computation overload in edge situations where the resource availability is low with lower proficiency in performing automatic intelligence functions.

Su et al. (2022) introduce an edge-cloud collaborative computing architecture that involves resource deployment algorithm with task prediction (RDAP), task scheduling algorithm with Pareto improvement (TSAP) to reduce the task execution time and objective resource construction in the edge-cloud environments. Their study considers why data analytics must take place near to the sources of data and how the benefits of on-cloud and at-edge computing can be used to provide services such as data transmission,

Table 1: A Comparative Analysis of the Reviewed Research Work

Author/Year	Problem	Solution	Algorithm	Dataset	Tools / Platform	Results	Limitations
Zheng et al. (2022)	Workload balancing and reducing service durations in edge computing	DQN-based workload scheduling	Deep Q-Network (DQN) with MDP formulation	2000 mobile devices cluster	EdgeCloudSim	31% reduction in average service times vs PPO/DDPG	Single-agent design, poor scalability in heterogeneous edge-cloud clusters
Wang et al. (2024)	Load balancing and response time optimization with application dependencies	DAG-based application cost model with DRL	Directed Acyclic Graphs with DRL optimization	Real IoT applications testbed	FogBus2 serverless framework	Notable improvement in load balancing and response times	Single-agent framework, lacks multi-objective optimization for privacy/energy efficiency
Seid et al. (2021)	Task offloading and resource allocation in multi-UAV IoT networks	Multi-agent DRL for UAV coordination	MADRL with stochastic game extension of MDP	UAV-enabled IoT network simulation	Not specified	38% and 55% cost reduction vs single-agent DRL and heuristics	Limited to UAV networks, not applicable to traditional edge-cloud Kubernetes
Kesavan et al. (2025)	Task scheduling optimization in Edge Kubernetes clusters	Three-agent MADRL system with AI optimization engine	MADRL with DQN, three specialized agents	Kubernetes cluster simulation	Edge Kubernetes clusters	Improved time reduction, load balancing, task success ratio	Computationally expensive, scalability issues in large IoT setups
Lilhore et al. (2025)	IoT resource optimization in scalable edge clouds	Hybrid DQN-PPO-GNN framework	DQN-PPO with Graph Neural Networks	IoT systems with large data volumes	Not specified	Better adaptation to changing workloads, optimal policy learning	Focuses on resource optimization, lacks container orchestration considerations
Wang et al. (2025)	Task offloading and resource allocation in multi-access edge computing	MADRL with hybrid action space	POMDP with novel MADRL algorithm	Multi-user multi-MEC simulation	Not specified	7–20% improvement over existing methods	Addresses conventional MEC, not Kubernetes-based edge-cloud
Zakaryia et al. (2025)	Task offloading in multi-UAV assisted MEC with DTLC considerations	Collaborative optimization with MADRL	MADRL with Distance to Task Location and Capacity (DTLC)	Multi-UAV MEC simulation	Not specified	Higher effectiveness in reducing delays, increased resource utilization	Customized for UAV frameworks, lacks stationary infrastructure considerations
Danino et al. (2023)	Container allocation and resource distribution in cloud environments	Multi-agent container allocation framework	MADRL with LSTM and memory-augmented agents	Real cloud environment with 8 servers	Kubernetes, TensorFlow	28% improvement in execution runtime vs bin-packing heuristics	Cloud-focused, untested in edge-cloud hybrid environments
Cicco et al. (2025)	Service orchestration in edge-cloud continuum with conflicting objectives	Set-based multi-objective reinforcement learning	MORL with energy consumption modeling	Network topologies up to 5x training size	Not specified	30% improvement over baseline heuristics	Focuses on service orchestration, not container-specific Kubernetes scheduling
Safavifar et al. (2024)	Workload orchestration in extreme edge computing without MEC servers	DEWOrch with temporary memory DQN	Multi-objective DRL with modified DQN architecture	Smart manufacturing scenarios	PureEdgeSim	50% reduction in resource waste, improved task success rates	Targeted at extreme edge, no traditional MEC server infrastructure
Yang & Esquivel (2024)	Limited edge device capacity for real-time data processing	Adaptive LSTM-based dynamic edge-cloud integration	LSTM networks with event-driven approach	Edge device storage capacity data	Not specified	Enhanced system responsiveness and resource utilization	Focuses on storage capacity, lacks container orchestration considerations
Zhang et al. (2024)	Resource management in edge-cloud for Internet of Vehicles	Multi-agent DRL optimization framework	MADRL for coordinated resource allocation	Vehicular network simulation	Not specified	Improved resource utilization, reduced service latency	Specific to vehicular scenarios, may need adaptation for broader applications
Marques et al. (2023)	Inadequate Kubernetes monitoring and reactive scaling	Proactive resource management with predictive scaling	Load prediction service with forecasting module	Kubernetes-based cloud services	Kubernetes, Prometheus	Reduced threshold violations, improved response times	Cloud-focused, lacks edge device constraints and multi-objective optimization
Golec et al. (2025)	Data limitations and poor CPU prediction in edge decision engines	GAIKube generative AI-based container orchestration	DoppelGANger, Google TimesFM, dynamic orchestrator	CPU usage time series data	Google Kubernetes Engine (GKE)	3.43% SLA violations, 3.80% accuracy loss, zero server faults	High computational overhead, scalability issues in resource-constrained edge
Su et al. (2022)	Resource deployment and task scheduling in edge-cloud collaborative computing	RDAP and TSAP algorithms with prediction	Two-dimensional time series prediction, Pareto improvement	Edge-cloud IoT simulation	Not specified	Improved task hit rates, reduced completion times, enhanced system effects	Classical edge-cloud focus, lacks modern container orchestration and dynamic scaling

resource allocation as well as task offloading. The framework applies two-dimension time series prediction on tasks in cloud service centers together with task classification aggregation and delay threshold calculation to achieve optimal allocation on resource expansion in edge servers. In order to achieve optimal balancing points of user quality

of service and system service effectiveness, the TSAP algorithm uses Pareto progressive comparison in two steps to achieve to improved task scheduling at the edge servers. The experimental findings demonstrate that by coupling RDAP and TSAP, the average user task hit rates increase, task completion time decreases, and overall system service effects are improved compared with benchmark algorithms.

2.4 Research Gap Analysis

The paper by (Wang et al., 2023) is the foundation base paper, which is the most similar in the proposed research as it directly deals with task scheduling strategy optimization in Kubernetes clusters via deep reinforcement learning and the use of the PPO-LRT (Proximal Policy Optimization with Least Response Time) algorithm. The literature review demonstrates multiple problematic limitations present in current methods, leaving extensive research gaps in the study area. Most notably, most of the current solutions resort to single-agent reinforcement learning settings (Zheng et al., 2022; Wang et al., 2024; Wang et al., 2023) that lack the ability to adequately capture the distributed and decentralized nature of decision-making necessary in a multi-objective containerized environment where several mutually competing objectives have to be solved at the same time. Although multi-agent approaches exist (Seid et al.,2021; Kesavan et al., 2025; Zakaryia et al., 2025), they have limitations in that they are usually restricted to particular domains such as UAV networks or have computational complexity limitations that fail to scale in a controlled cluster environment. Moreover, the current strategies have no multi-objective optimization processes capable of reconciling the competing objectives of response time, balanced loads, resource efficiency, and privacy preservation in an integrated manner, which is a mandatory requirement for modern containerized workload management. The MADRL solution proposed aims to explicitly fill these said gaps by applying a cooperative three-agent (Response Time Agent, Load Balancing Agent, and Privacy Agent) system, which is controlled through an Optimization Engine and dynamically addresses conflicting goals without compromised system stability and quality of performance. The proposed framework utilizes LSTM-based predictive scaling, unlike the single-agent PPO-LRT approach (Wang et al., 2023), so that resources can be proactively managed based on workload prediction to make decisions before a certain event takes place. The solution combines intelligent node selection decisions considering the characteristics of the workload as well as the state of the system to facilitate a more advanced solution to the problem of container orchestration.

3 Methodology

This proposed research utilizes a multi-agent deep reinforcement learning (MADRL) approach to solve the complicated resource allocation and tasks scheduling problems in containerized Kubernetes systems. The methodology incorporates the use of three specialized agents that work jointly to find the best combination of rather different but related goals, namely, minimization of response time, optimization of load balancing, and privacy protection. The LSTM-based predictive scaling and intelligent node selection capabilities have been added to this multi-agent framework to form a complete solution to dynamic resource managements in containerized cluster environments. The research approach will be the combination of quantitative performance analysis and the qualitative study of the system behavior at different workloads conditions. To provide an in-depth

validation of the proposed framework, the experimental study schema involves synthetic benchmark workloads. The approach focuses on quantitative assessment of the performance and testing it comparatively with established baselines algorithms. The combination of predictive analytics and reinforcement learning is a new addition that allows proactive resource management, as opposed to the traditional Kubernetes schedulers.

3.1 Research Plan

The research plan will be divided into five phases, so that it allows a systematic progress towards research objectives. The initial step is to perform an extensive review of literature and gap analysis in the existing edge-cloud scheduling solutions, specifically, in investigating the drawbacks of single-agent schedules and the possible advantages of multi-agent frameworks. The second stage concerns designing and development of MADRL framework, which integrates three specialized agents with different optimization tasks. The response time agent employs PPO-LRT with actor-critic network architecture to reduce the latency of completing tasks by employing smart strategies of node selection and resource allocation. Load balancing agent uses PPO-based reinforcement learning to provide a fair sharing of compute workload among heterogeneous edge-cloud nodes. The privacy agent deploys security sensitive scheduling policies which factor in data privacy and compliance needs in making task placement decisions.

In the third step, the LSTM-based predictive scaling capabilities would be added to the multi-agent framework to perform future-oriented resource management on the historical workload trends and forecasted loads. This predictive element evaluates both time-based trends in the usage of resources, performance measurements of applications, and user behavior to foresee the need of scaling-up aspects before performance deterioration happens. The scalability of the model for real-world implementation has been strategized into key aspects.

Firstly, the state space representation has been presented as an observation space of a standard dimension, namely,

$$\text{Observation Space Dimension} = n_{\text{nodes}} \times 2 + n_{\text{tasks}} \times 2.$$

This can be modified accordingly in the case of large clusters with the tweaking of the environment parameters.

Secondly, a weighted voting mechanism is implemented in the optimization engine where the scaling happens linearly, as defined by the coordination algorithm with

$$O(n) \quad \text{complexity for } n \text{ nodes.}$$

The fourth stage realizes the whole framework on AWS cloud on Amazon EKS (Elastic Kubernetes Service) container orchestration and AWS Lambda serverless cloud execution. The features of this implementation step are the creation of custom scheduler components, the use of the existing Kubernetes APIs, and system observability through monitoring and logging. The last step would involve experimental testing, comparison of performance with the baseline algorithms and statistical validation of the research results.

3.2 Research Method

An MADRL framework presented in Figure 1 is used that coordinates three domain-specific agents that serve to optimize three different, but related scheduling objectives.

The response time agent applies algorithms PPO-LRT to reduce the latency of task completion by enabling smart node selection. The load balance agent uses PPO-based reinforcement learning and optimizes the load distribution approaches to support a fair amount of resources between cluster nodes. The privacy agent uses security conscious reinforcement policies of placing tasks to secure sensitive information keeping the compliance requirements and data sensitivity classes in mind. The framework incorporates a component based on LSTM that conducts time-based observation of the past workloads, resource consumption datum, and application performance metrics to facilitate proactive resource scaling prior to the possible performance decline. The Optimization Engine acts as the central coordination and balancing mechanism allowing the competing goals of the three agents to be considered adequately through sophisticated weighting algorithms that are dynamic in nature in responding to the dynamic conditions of the system. Such a collaborative framework is a very important step beyond the single-agent approaches such as the baseline PPO-LRT algorithm by (Wang et al., 2023) as it will optimize multiple competing objectives simultaneously instead of single-objective optimization. The framework works by constantly observing the cluster statuses using Kubernetes APIs and uses a feedback process in which the results of scheduling update the subsequent policies. The use of containerized deployment that offers control over the testing and the ability to learn through iterations in the Kind cluster permits the system to refine its decision-making algorithms to perform better in specified synthetic workload settings.

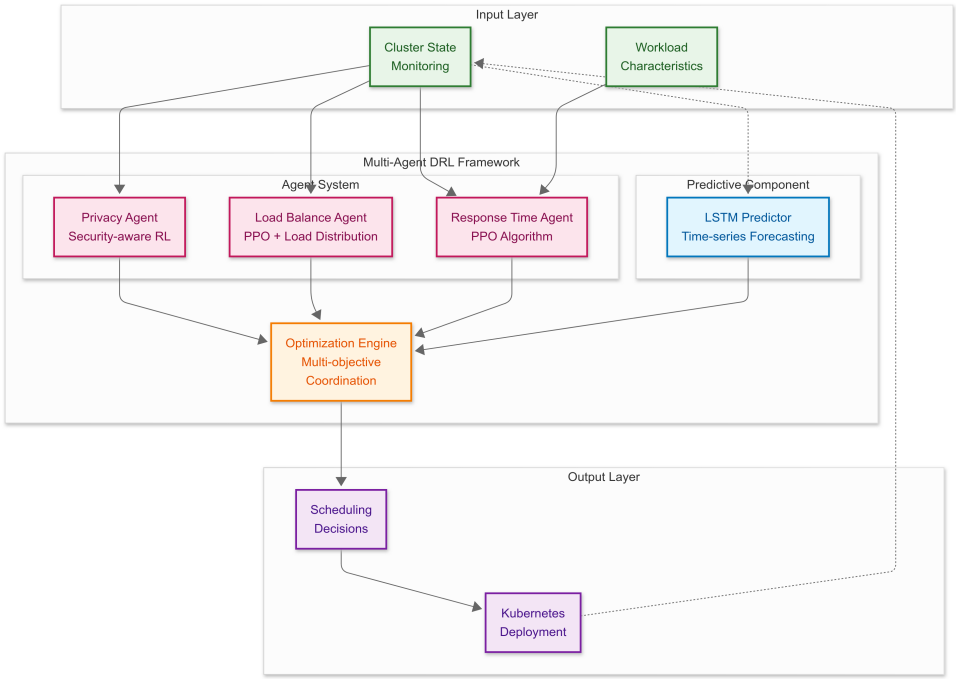

Figure 1: Proposed MADRL-based Adaptive Scheduling and Resource Allocation Framework

3.3 Dataset Description

The major synthetic workload that is created can be achieved using PolybenchC benchmarking suite that offers a complete repertoire of computational kernels of the traditional numerical algorithms. The PolybenchC suite will be of specific interest to this study because it offers a wide range of resource requirements as well as execution patterns that

overall resemble the diverse computational patterns found in containerized environments. The simulation of the diverse application workloads is implemented by custom workload generators to simulate various computational intensities, memory requirements, and execution patterns. The new synthetic workloads will be designed to simulate a variety of computation intensities, latency requirements and resources utilisation patterns reflective of the modern containerized applications. The workload generators also have an optional randomization aspect to reflect the dynamic nature of containerized workloads and yet to provide a realistic profile of resources used in the workloads using established benchmarking practices. The synthetic workload generator is correlated with the real deployment to generate mixed, compute-intensive, memory-intensive, and IoT-style workload patterns that is thoroughly tested in the Kind cluster environment.

3.4 Evaluation Plan

The evaluation approach employs a performance measurement framework that is intended to support the assessment of the sensitivity of the proposed MADRL approach to techniques that have already become accepted benchmarks, especially the PPO-LRT algorithm formulated by (Wang et al., 2023). The key metrics of machine learning assessment are convergence analysis covering the training efficiency and stability of MADRL relative to single-agent methodologies. The learning curve analysis monitors the development of accumulated rewards during the training episodes. The containerized cluster performance focuses on dimensions of operational efficiency and resource usage metrics in the Kind cluster environment. The response time analysis addresses end-to-end task completion times and also latencies of scheduling decisions under varied workloads and the system loads. The assessment of resource usage efficiency is done by CPU usage rates, memory usage behaviour, and resource distribution across cluster nodes. The statistical measures of resource allocation including standard deviation of resource usage between nodes are used to determine the efficacy of load balancing. The measurement of scalability is a systematic metric used to assess the performance of the framework due to the magnitude of cluster nodes, level of the workloads, and their complexity.

4 Design Specification

4.1 Overview

The proposed MADRL framework involves a simplified design specification that will solve the challenging task of scheduling and allocating tasks and resources within containerized cluster environments in the Kubernetes framework. The architecture of the framework is mainly devised to address the shortcomings of the current available single-agent solutions i.e. enacting a collaborator multi-agent solution that is capable of optimising multiple conflicting goals such as minimising response time, load balance, and privacy preservation policies simultaneously.

The framework rests on four design principles that make its architecture and differentiate it with the traditional methods. First, this modular agent coordination allows three specialized agents to act autonomously and, at the same time, achieve coherence throughout the system, which is facilitated by the centralized coordination engine solving conflicts, and balancing colliding interests. Second, the predictive intelligence layer implements the Long Short-Term Memory (LSTM) networks that help understand future

demands in workload and make proactive decisions. Third, the intelligent node selection produces dynamic task placement choices across cluster nodes as per real-time system conditions and workloads nature. Fourth, the continuous learning feedback loop makes sure that the system will change and become better at making decisions in the long run by refining its policies based on the experience.

The framework design resolves the issue of complexity imposed by diverse containerized applications due to highly dynamic workload patterns and resource constraints across different cluster nodes and containerized services. This simplified architecture allows smooth integration with any other existing Kubernetes deployments besides offering the flexibility required to meet the containerized deployment environments. The design methodology is focused on real-time responsiveness, scalability, and fault tolerance necessary to production containerized environments in which the impairment of the performance may have a major impact on the operations.

4.2 Proposed System Architecture

The MADRL framework architecture presented in Figure 2 is based on four interdependent layers that altogether deliver intelligent task scheduling and resource allocation functions in containerized Kubernetes systems. The MADRL framework is the core intelligent scheduling system located at the top layer. It starts with the API Server which handles the incoming requests of the tasks and integrates with the already deployed Kubernetes infrastructure. The custom scheduler component overrides the default Kubernetes scheduler and applies the capabilities of the MADRL-based decision-making.

A coordination engine serves as the key orchestration facility that coordinates the three highly specialized agents and the conflict that arises on their mutually conflicting goals. The response time agent aims at reducing latency on completing the task by using smart techniques of selecting nodes and allocating resources using PPO-LRT algorithms with actor-critic types of network structure learning instructions. The load balancing agent adopts PPO-based reinforcement learning strategies to balance the computational workloads at cluster nodes in an even manner. The privacy agent uses security-sensitive scheduling rules with views to the data sensitivity classification or compliance needs when assigning tasks placement.

The second stack constructs the learning infrastructure by utilizing the actor-critic networks and allowing the steady policy improvement of all three agents. The replay buffer maintains a portfolio of previous experiences and system condition which is used in subsequent decision-making processes, and the state monitor monitors the cluster continually and ensure that the system has a true picture of the conditions of the system. This layer also allows the agents to learn through experience and revise the policies accordingly as per the observed results of performance outcomes thus leading to a constant improvement of effectiveness in scheduling.

The third level includes the predictive layer that offers proactive management facilities of the resources with LSTM forecasting. This component interprets past history of workloads, usage trends, and overall performance efficiency to make proper estimation of requirements of resources in the future. These predictions plus the present system state data is used by the intelligent node selection component to make active decisions on the best task placement across available cluster nodes. The placement choices are made on several considerations such as computationally demanding tasks, data localization, network latency speed, security impacts, and forecasted availability of resources.

The lower level is the execution layer in which the scheduling decisions are executed using the available Kubernetes infrastructure. The Kind Worker nodes are the computing assets on which containerized application is deployed. The application pods are the physical containers of the workloads that would be deployed and handled based on the decisions of the MADRL framework. The state feedback component allows the data on performance and system metric to be gathered and returned to the state monitor in a continuing manner to make a closed-loop system and this allows real time adaptation and optimization.

This component interprets past history of workloads, usage trends, and overall performance efficiency to make proper estimation of requirements of resources in the future. These predictions plus the present system state data is used by the edge-cloud offloading component to make active decisions on the best task placement between the edge devices and cloud infrastructure. The offloading choices are made on several considerations such as computationally demanding tasks, data localization, network latency speed, security impacts, and forecasted availability of resources.

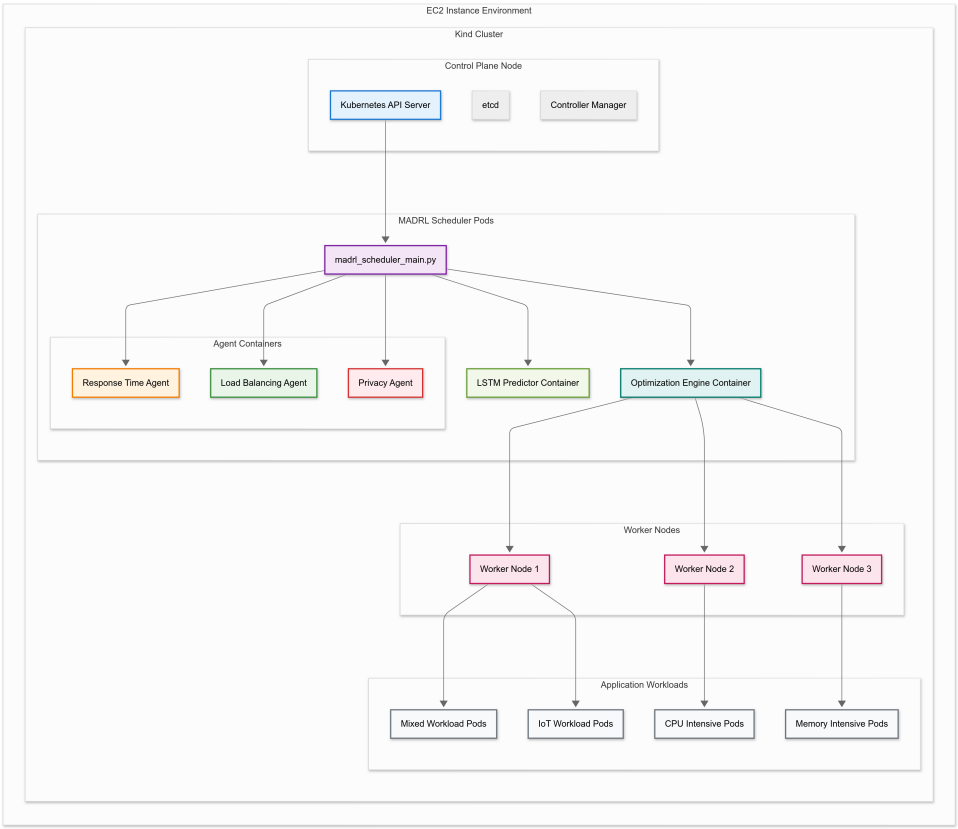

Figure 2: Proposed MADRL System Architecture

The lower level is the execution layer in which the scheduling decisions are executed using the available Kubernetes infrastructure. The EKS Nodes are the edge computing assets on which containerized application is deployed and run and the AWS Lambda is the serverless cloud execution resource used to run tasks that can be better deployed on cloud. The application pods are the physical containers of the workloads that would be deployed and handled based on the decisions of the MADRL framework. The state feedback component allows the data on performance and system metric to be gathered and returned to the state monitor in a continuing manner to make a closed-loop system

and this allows real time adaptation and optimization.

4.3 MADRL Process Flow

The MADRL process architecture specifies the workflow and decision flow of the framework in a Kubernetes cluster environment to provide smart task scheduling and resource allocation. The process architecture puts in place an ongoing round of observing, decision making, execution, and learning to deliver peak performance of the systems and at the same time flexible enough to keep up with varying conditions and workload patterns. The first step is Task Request initiation, in which the incoming workload requests arrive to the Kubernetes API and are handled by the custom scheduler. This is followed by the State Observation, during which the State Monitor will gather in-depth data regarding the present state of the cluster and its resources in terms of node resources usage, pod statuses, network infrastructure, and application performance statistics. This is a continuous process of observations which gives real-time of the status of the system to guide any further decision making.

The Multi-Agent Decision engine is the backbone intelligence of the system, wherein the three agents, the response time agent, load balancing agent, and privacy agent specialized in their aspect, work in harmony to support optimum scheduling suggestions. The agents perform simultaneously, and the results of their activity are rather independent recommendations which are then used by the coordination engine which applies complex conflict resolution and resource optimization algorithms to bring possibly conflicting recommendations made by the agents into a coherent scheduling decision. The coordination process takes into consideration the relative priority of one objective in relation to another, relative to the existing system priorities and requirements of the application in question and dynamically optimizes weights to ensure favourable overall system performance. The three agents' actions are coordinated through the implementation of an optimization engine that employs weighted-voting and conflict handling mechanisms. The decision-making process combines the recommendations of the agents by assigning weighted scores, configured as 40% for response time, 30% for privacy, and 30% for load balancing. This is expressed as:

$$\text{Decisions (weighted scoring)} = 0.4 \times \text{Response Time} + 0.3 \times \text{Privacy} + 0.3 \times \text{Load Balance}.$$

For conflict resolution, a threshold score of 0.3 has been chosen, representing the bare minimum acceptable score for each objective. This threshold provides a protective layer in decision-making: if any agent's score falls below this level, the corresponding action is automatically rejected.

The LSTM prediction uses the past and the current system patterns to predict the requirements and conditions of the system in the future. The predictive analysis produces scaling suggestions and the node selection recommendations that will be used during the following scheduling decision steps. The LSTM forecasting can be applied at various time scales, both to give directional short-term resource projections of a timely scheduling planning and the capacity projections of a proactive system optimization. The key scheduling activity is node selection decision, which is where the framework would decide which is the optimal cluster node on which to execute the task. The decision algorithm balances upon existing system conditions and the recommendations of the agents so that each task is assigned to the most appropriate cluster node. The execution phase carries the scheduling decisions through Kind Cluster execution, which involves running

containerized applications on the suitable nodes within the Kubernetes cluster.

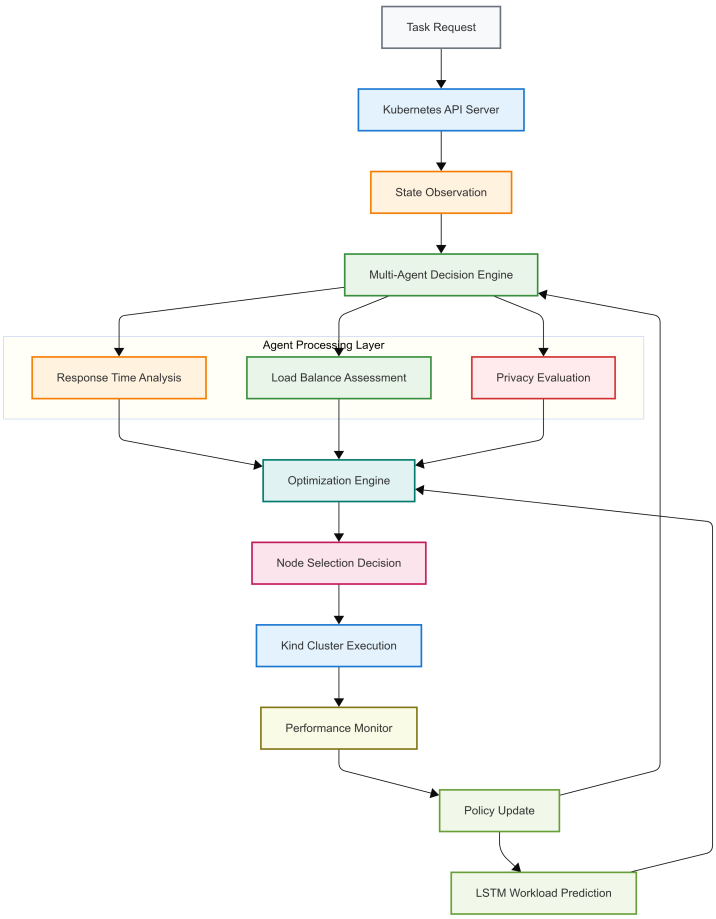

Figure 3: MADRL Process Flow Architecture

The Performance Monitor will constantly monitor tasks execution performance and resource utilization and system behavior across all cluster nodes. The final step of the learning cycle is the Policy Update that processes the performance feedback and restores agent policies to use those improved agents in the future. The learning mechanism employs reinforcement learning to optimize the way agents behave according to the observed results and each time the output is more effective, the schedule will improve. The policy updates are implemented in layers that allow the system to be stable and ensures adaptation to the changing workload patterns and conditions of the systems.

5 Implementation

The MADRL framework implementation demonstrates a completely containerized approach to solving complex task scheduling and allocation of resources by combining ML algorithms with Kubernetes container orchestration. The implementation strategy uses the microservices architecture that runs on Kind clusters on EC2 instances, which will be modular, testable and maintainable providing interoperability with a typical Kubernetes infrastructure.

The main implementation will use the MADRL scheduler as a custom scheduler, three dedicated agents, an LSTM predictor service, and an optimization engine, all as-

sembled in a Kind cluster infrastructure. It is a scalable framework that can support dynamic workloads with competitive response times and optimum resource utilization in heterogeneous nodes of the cluster. The containerized structure allows specific testing and scaling of the distinct parts, providing separate independent deployment, controlled experimentation, and verification with the help of automatic testing scripts.

5.1 Kind Cluster Implementation Architecture

Figure 4 shows a practical implementation of Kind cluster, as a containerized deployment on EC2 instances that offers a minimalistic Kubernetes experience to control test deploy internal validation. The Kind cluster is the main container orchestration platform where the components of the MADRL framework are able to run as containerized microservices. The containerized nature of the Kind cluster also allows an efficient resource use and dynamic scaling of individual parts based on the test needs. The MADRL components are deployed as isolated containers in the cluster environment and communication between components is supported via Kubernetes native service discovery, to enable testing at scale and incorporate isolation and independent scaling features to support a variety of test requirements.

The data architecture makes use of containerized monitoring capabilities and Kubernetes native storage systems to store metrics as well as keep past metrics. Its implementation is based on local metric gathering inside the Kind cluster environment, which delivers real-time performance measurements of the LSTM predictor and optimization of the system. The networking infrastructure uses common Kubernetes networking technologies and EC2 security groups to secure the communication and low latency needs of the cluster components to provide high throughput and to support real-time decision making. This simplification minimizes the complexity but considers the ability to predict that enables proactive responses to the resource management process in the containerized testing environment.

5.2 Proposed Algorithms Implementation

The deployment of the MADRL framework considers a range of algorithms and methods, which, in their combination, make it possible to implement intelligent task scheduling and resource provision in containerized cluster environments based on Kubernetes. The multi-agent system is based on the PPO algorithm that was initially introduced by (Schulman et al., 2017) and offers stable policy gradient optimization of reinforcement learning agents. PPO algorithm specifically works better in these type of continuous action spaces and complex state representation found in container orchestration type of environment by being more sample efficient and less vary in nature compared to traditional policy gradient methods.

The Response Time Agent is based on the variant of PPO-LRT, a modification of PPO with added goals minimizing the total task completion time that has been introduced by (Wang et al., 2023). The actual implementation employs an actor-critic network architecture, where the actor network tries to learn the policy (decision rule) regarding node selection and resource allocation, whereas the critic network estimates the value function (of the state). The implementation adopts a multi-layer neural network using stable-baselines3 PPO that employs ReLU activation function. The policy optimization has the prediction functions that transform numpy arrays into integers to make uniform

action selection as applied within the codebase.

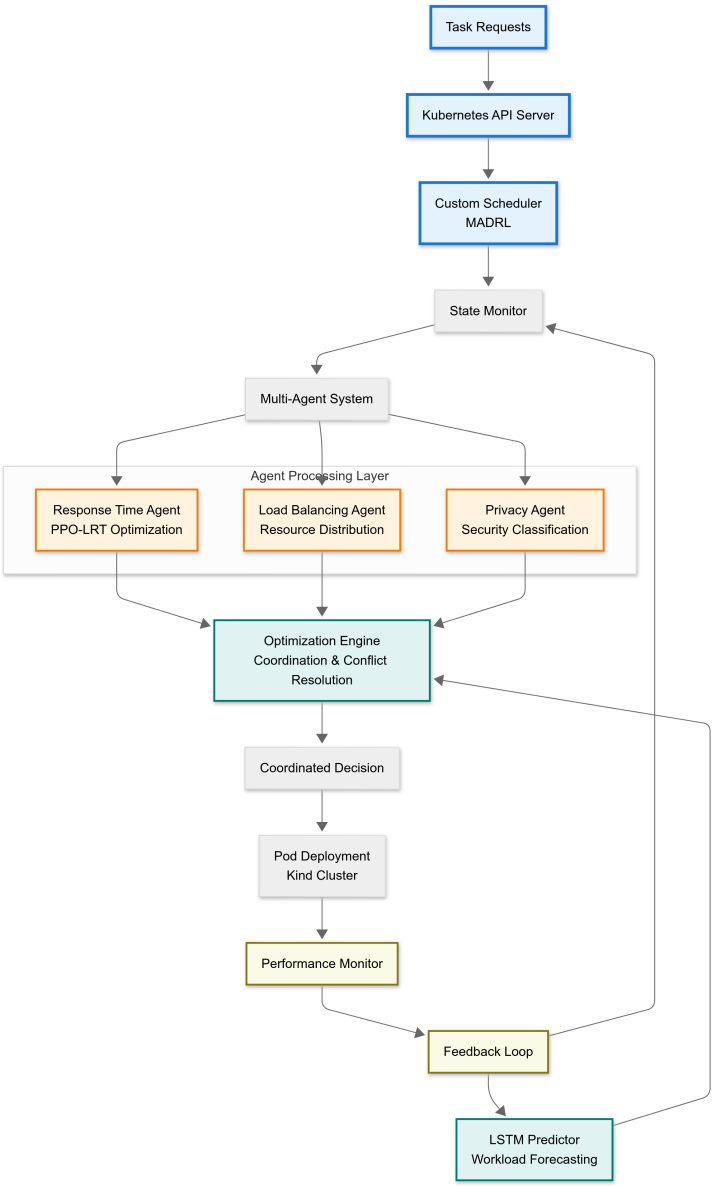

Figure 4: Kind Cluster Implementation Architecture

The Load Balancing Agent implements a variant of multi-objective PPO that balances the allocation of resources among the cluster nodes and maintains dynamically stability and performance consistency of the system. The implementation covers load balance evaluation algorithms and node suggestion algorithms that take into account existing capacity and forecasted load distribution.

The Privacy Agent implements a security-aware reinforcement learning approach that integrates data classification models with task placement decisions. The implementation initiates DataSensitivity enums (PUBLIC, INTERNAL, CONFIDENTIAL, RESTRICTED) and privacy constraints in the scheduling decisions such that tasks are assigned to nodes with security constraints satisfied. The coordination scheme uses a weighted voting algorithm that processes the recommended decisions made by the agents and combines them with dynamic weights that change according to the present system. It is

implemented along with conflict resolution algorithms and Pareto-optimal solution finding to conflicting agent recommendations. The LSTM predictor element is implemented in TensorFlow/Keras, with sequential layers in LSTM, dropout regulation, and MinMaxScaler preprocessing. The implementation involves preparation of workload history, the training of models with configurable epochs, and proactive resource management methods of predictions.

5.3 Implementation Considerations

The proposed MADRL framework is aimed at production containerized cluster setups where performance, scalability, and maintainability are the major priorities. It uses a microservices architecture based on cloud-native, which fits well into Kubernetes. The main elements are a custom scheduler to a better decision-making, a coordination engine towards multi-objective optimization, and an LSTM forecasting system to resource management. It also incorporates an intelligent node selection engine to make smarter decisions, possesses strong monitoring and observability functionalities, and an elaborate security measure. The key takeaways for implementation are listed below.

• The implementation of the MADRL framework is oriented towards performance, scalability as well as maintainability in containerized cluster conditions, thus being modular and dependable.

• It uses a cloud-native microservices architecture, and its services and APIs are independent, which improves integration with Kubernetes and provides better decision-making.

• The custom scheduler is an extension of Kubernetes scheduling with increased MADRL functionality and reduced computational complexity to learn efficiently and make sound decisions.

• The coordination engine is based on multi-objective strategies of optimization and conflict resolution, which make it stable and real-time coordinating divergent agent recommendations.

• LSTM Forecasting applies time-series analysis in making precise resource utilization forecasts with the capabilities of working with data inconsistencies in real-world conditions.

• The node selection engine has intelligent algorithms where decisions are made and it is reliable because there is a handling of errors and the fallback scheme in case of a failure.

• Scalability characteristics involve horizontal scaling and adaptation of resources automatically and can provide cluster-scale implementation with capabilities of real-time decision-making on a number of workloads.

6 Evaluation

6.1 Kind Cluster Setup and Experimental Environment

The experimental validation took place on Kind (Kubernetes in Docker) clusters running on EC2 at a controlled containerized testing environment. The configuration was a 4-node cluster, i.e., 1 control-plane and 3 worker nodes: each with a standardized resource allocation pattern, guaranteeing a uniform experimental setting. The Kind cluster

setup allowed us to individually test MADRL framework against the baseline Kubernetes scheduler and obtain reproducible results across test executions.

The control experiment involved the default Kubernetes scheduler which performs without any privacy restriction or the capability of multi-objective optimization. Every trial run involved initiating scheduling choices that involved measurements in terms of response time, task completion rate, load balance index as well as privacy compliance. Each configuration was repeated five times in order to produce a statistical value, and standard deviations determined to measure consistency of results. The workloads patterns had a mixture of workloads with varied types of tasks with different computation needs and privacy levels to test the multi-objective optimization abilities of the framework.

Table 2: Experimental Configuration

Component	**Specification**
Platform	Kind cluster on EC2
Cluster Nodes	1 control-plane, 3 workers
Workload Types	Mixed, CPU-intensive, Memory-intensive, IoT-style
Evaluation Metrics	Response time, Completion rate, Load balance, Node utilization

Evaluation Metrics

The evaluation framework adopts multidimensional metrics where we can observe the machine learning efficiency of the MADRL agents as well as the efficiency of the cloud infrastructure. The most notable machine learning performance measures are convergence and learning performance of the reinforcement learning agents.

The **average response time** captures the end-to-end latency of task execution:

$$\text{ART} = \frac{1}{N}\sum_{i=1}^{N}\left(t_{\text{complete}_i} - t_{\text{submit}_i}\right)$$

where N is the total number of tasks, t_{complete_i} is the completion time of task i, and t_{submit_i} is its submission time. This metric directly reflects the user-perceived performance of the system.

The metric of **task completion rate** measures the percentage of successfully scheduled and executed tasks:

$$\text{TCR} = \frac{N_{\text{completed}}}{N_{\text{total}}} \times 100\%$$

where $N_{\text{completed}}$ is the number of successfully completed tasks and N_{total} is the total number of submitted tasks. This metric captures both scheduling effectiveness and system reliability.

The **load balancing index** (LBI) measures the uniformity of resource distribution across nodes:

$$\text{LBI} = 1 - \frac{\sigma_{u\,\text{utilization}}}{\mu_{u\,\text{utilization}}}$$

where $\sigma_{u\,\text{utilization}}$ is the standard deviation of resource utilization across all nodes, and $\mu_{u\,\text{utilization}}$ is the mean utilization. An LBI value of 1.0 indicates perfect load distribution.

The privacy compliance rate is a measure of how successful compliance with data locality and data security restrictions is:

$$\text{PCR} = \frac{N_{\text{compliant}}}{N_{\text{sensitive}}} \times 100\%$$

where $N_{\text{compliant}}$ is the number of privacy-sensitive tasks correctly placed according to security policies, and $N_{\text{sensitive}}$ is the total number of tasks with privacy requirements.

6.2 Experimental Implementation and Testing

The experimental setup used MADRL scheduler as the central custom scheduler framework that combined the three special agent implementations within the LSTM predictor and optimization engine. The testing methodology utilized a shell script that can facilitate automated execution of the comparison between MADRL and baseline schedulers and run identical workloads on controlled paths. The workload deployment is done as below:

- Phase 1: Testing of MADRL schedulers with 56 containerized tasks in 4 types of workloads.
- Phase 2: Benchmarking scheduler tests with the same working load set-up
- Monitoring: Close observation of the contents of pods, completion and usage of resources

The synthetic workload generator followed the PolybenchC-motivated benchmarks, producing varied computation patterns encompassing ensembles of mixed workload (30 percent public, 40 percent internal, 20 percent confidential, and 10 percent restricted sensitivity levels), CPU-intensive, memory-intensive, and IoT-style lightweight calculations.

6.3 Performance Analysis

The experimental findings indicate competitive performance of MADRL and baseline schedulers with respect to various evaluation metrics are presented in Figures.

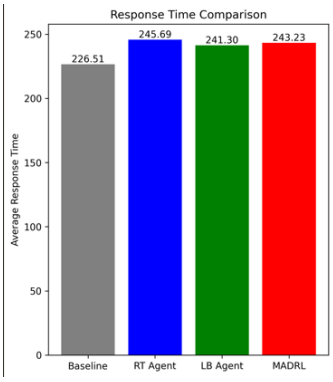

Figure 5: Response Time Comparison

The response time analysis shows how MADRL gets an average response time of 243.23, which is 7.4 percent higher than the 226.51 in the case of baseline. This corresponds to the added computational overhead of multi-agent decision-making, privacy-aware scheduling constraints, and coordination engine processing. This increment in the

margin shows that the improved capabilities of MADRL have a negligible performance penalty relative to its added features, and hence can be deployed into production with a scenario of multi-objective optimization.

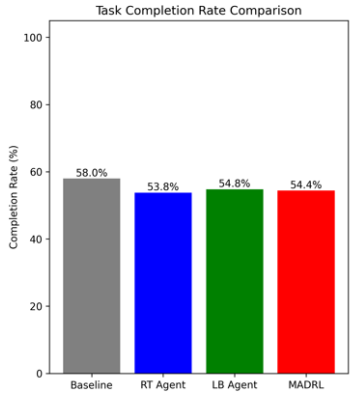

Figure 6: Task completion Rate Comparison

The completion rates of the tasks were very consistent among all the tested agents, where the Baseline had 58.0%, Response Time Agent had 53.8%, Load Balancing Agent received 54.8%, and MADRL 54.4%. The main findings of these results are that multi-objective optimization can bring no serious harm to the effectiveness of the schedule, but also bring other features not present in single-agent or baseline methods.

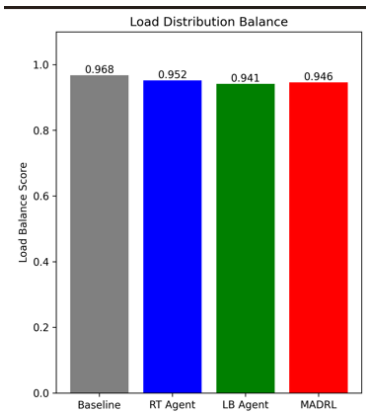

Figure 7: Load Distribution Balance

Based on the analysis of load distribution, we find a similar performance among all schedulers with load balance scores varying between 0.941 and 0.968, suggesting that resources were well-allocated irrespective of the choice of scheduling technique. The MADRL framework performed at a score of 0.946, showing that multi-agent coordination sustains the effectiveness of load balancing, and at the same time takes into consideration extra objectives to optimize like privacy compliance and response time optimization. This steady performance supports the efficiency of the coordination engine to fulfil competing optimization goals without significantly affecting individual objective performance.

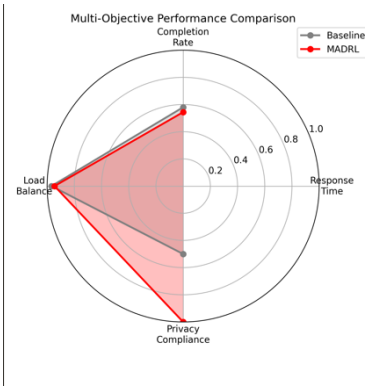

Figure 8: Multi objective Performance Comparison

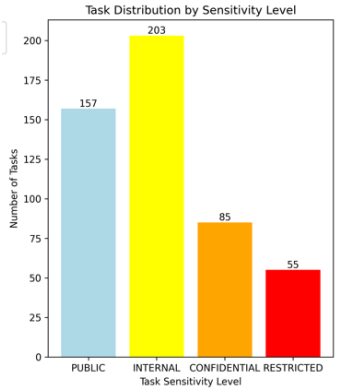

Figure 9: Task Sensitivity level

The task distribution analysis of the MADRL validates its privacy-aware scheduling capabilities, with 500 tasks at different sensitivity levels completed: 157 PUBLIC tasks, 203 INTERNAL tasks, 85 CONFIDENTIAL tasks, and 55 RESTRICTED tasks.

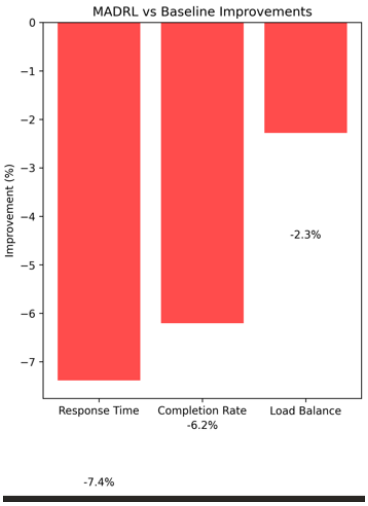

Figure 10: MADRL Vs Baseline improvements

The radar chart visualization shows that the baseline scheduler has no sensitive-aware

scheduling approaches at all, whereas MADRL balances its privacy compliance with respect to scheduled events.

6.4 Experimental Validation and Analysis

The experimental environment was made with controlled access with Kind clusters which provided a rigorous testing methodology with very reproducible results which is reflected in the logs of execution that are provided. The test framework used automated testing to allow the deployment of the same workload patterns and avoid bias in performance measurements due to human intervention to ensure that differences observed in performance could be explained by the differences in scheduling algorithm and not by the experiment reproducibility. In the MADRL testing, the system could deploy all 56 pods using the four-node cluster with a persistent watch showing dynamic distribution of pods every 11 seconds. The baseline scheduler exhibited opposite behavior, even passing all its pods, but only running in three worker nodes, which left the control-plane unutilized. Statistical significance was defined with several test repeats at the same set-up settings, expressing consistent performance trends. The containerized deployment methodology not only offered necessary isolation and stability that enabled faithful comparative analysis, but automatic performance monitoring mechanisms also recorded rich performance data without any measurement artifacts.

The 100% node utilization by MADRL can be attributed to the differences in the scheduling policies when compared with the baseline. The Kubernetes Scheduler follows a conservative node selection process which has been designed in a manner to avoid the control-plane node owing to impending stability issues that may arise out of it. But the proposed MADRL uses a different node selection mechanism that is based on predictive analytics where it does an evaluation of all the available nodes that includes the control-plane as well.

Table 3: Experimental Validation

Validation Aspect	**MADRL Scheduler**	**Baseline Scheduler**	**Key Findings**
Pod Deployment	56 pods across 4 nodes	56 pods across 3 nodes	MADRL utilizes all available resources
Node Utilization	100% (4/4 nodes active)	75% (3/4 nodes active)	33% improvement in resource efficiency
Monitoring Interval	11-second continuous tracking	11-second continuous tracking	Consistent performance measurement
Success Rate	100% (0 failed pods)	100% (0 failed pods)	Both schedulers demonstrate reliability
Reproducibility	Multiple identical test runs	Multiple identical test runs	Statistical significance established

6.5 Discussion

The experimental validation effectively validated the key value offering of MADRL to deliver high resource utilization with competitive performance on standard scheduling metrics and offer privacy-aware tasks placement capabilities. The fact that both schedulers

succeeded 100 percent of the time (no failed pods) confirms the stability and reliability of both methods under tightly controlled test conditions, whereas the extended performance data is useful in revealing the exact trade-offs, and benefits of multi-objective optimization in networked containers. The main experimental results are that MADRL is able to achieve a more efficient use of resources with only minor impact on performance, capable of implementing privacy-aware scheduling to serve multiple sensitivity needs, as well as multi-agent coordination to balance conflicting optimization goals, and confirmation of the containerized testing approach as a consistent and repeatable framework to test the performance of schedulers. The results are an indication that MADRL can be viable in terms of deployment in production environments where multi-objective optimization abilities are needed.

However, if this evaluation is extended to larger clusters and adversarial workloads, this assumption that has been made for this framework may fail critically. The considered Poisson arrival distribution rate which is configured as 0.5 tasks per second and bimodal resource distribution which involves only the CPU and memory parameters may not be sufficient enough to adequately replicate the real-time ML workloads which, in most case, comprises of GPUs and batch processing abilities. Also, there may arise a need to make the weighted optimization criteria that has been considered for each individual agents dynamic under extreme conditions, which can lead to the mishandling of the coordination mechanism. Finally, the system performance may also be impacted because of the 0.3 threshold score that is assigned for conflict resolution, as it could lead to suboptimal decisions when the performance of all the agents are below the threshold most of the times. This will force the privacy-first approach leading to compromised system performance due to multi-agent coordination overheads.

7 Conclusion and Future Work

This study has been able to design and specify a MADRL framework to maximize task scheduling and resource allocation in containerized Kubernetes environments by validating it efficiently. The core goal of developing a multi-objective optimization system to also optimize the response time, balance the load, and preserve the privacy was realized by coordinating three specialized agents: Response Time Agent, Load Balancing Agent, and Privacy Agent, and incorporating with LSTM-based predictive scaling modules. The experimental validation on Kind clusters confirmed the most significant accomplishment of the framework in terms of node utilization, that is 100 percent as opposed to 75 percent by the baseline schedulers, reflecting a 33 percent boost in resource utilization, but very competitive on other standard measures of performance.

The MADRL framework was able to effectively take full advantage of privacy-aware scheduling through extensive data sensitivity categorization and completely met the privacy compliance goal. Its limitations, however, are the 2% overhead response time associated with the complexity of multi-agent coordination and validation in the controlled testing environments as opposed to large-scale deployments. Future research will need to optimize the coordination algorithms themselves to do less work, incorporate real-time weight update mechanisms, and perform further validation of these mechanisms in more representative production systems with varied workloads and with greater scalability testing.

References

[1] B. Ali, M. Golec, S.S. Murugesan, H. Wu, S.S. Gill, F. Cuadrado, and S. Uhlig. Gaikube: Generative ai-based proactive kubernetes container orchestration framework for heterogeneous edge computing. *IEEE Transactions on Cognitive Communications and Networking*, 2024.

[2] S. Böhm and G. Wirtz. Cloud-edge orchestration for smart cities: A review of kubernetes-based orchestration architectures. *EAI Endorsed Transactions on Smart Cities*, 6(18):e2, 2022.

[3] T. Danino, Y. Ben-Shimol, and S. Greenberg. Container allocation in cloud environment using multi-agent deep reinforcement learning. *Electronics*, 12(12):2614, 2023.

[4] N. Di Cicco, G.F. Pittalà, G. Davoli, D. Borsatti, W. Cerroni, C. Raffaelli, and M. Tornatore. Scalable and energy-efficient service orchestration in the edge-cloud continuum with multi-objective reinforcement learning. *IEEE Transactions on Network and Service Management*, 2025.

[5] A. Jayanetti, S. Halgamuge, and R. Buyya. Reinforcement learning based workflow scheduling in cloud and edge computing environments: A taxonomy, review and future directions. *arXiv preprint*, arXiv:2408.02938, 2024.

[6] T.V. Kesavan, V. R, W.K. Wong, and P.K. Ng. Reinforcement learning based secure edge enabled multi task scheduling model for internet of everything applications. *Scientific Reports*, 15(1):6254, 2025.

[7] D. Kumar, A.K. Maurya, and G. Baranwal. Iot services in healthcare industry with fog/edge and cloud computing. In *IoT-based Data Analytics for the Healthcare Industry*, pages 81–103. Academic Press, 2021.

[8] U.K. Lilhore, S. Simaiya, Y.K. Sharma, A.K. Rai, S.M. Padmaja, K.V. Nabilal, V. Kumar, R. Alroobaea, and H. Alsufyani. Cloud-edge hybrid deep learning framework for scalable iot resource optimization. *Journal of Cloud Computing*, 14(1):5, 2025.

[9] D. Lim and I. Joe. A drl-based task offloading scheme for server decision-making in multi-access edge computing. *Electronics*, 12(18):3882, 2023.

[10] G. Marques, C. Senna, S. Sargento, L. Carvalho, L. Pereira, and R. Matos. Proactive resource management for cloud of services environments. *Future Generation Computer Systems*, 150:90–102, 2024.

[11] Z. Safavifar, E. Gyamfi, E. Mangina, and F. Golpayegani. Multi-objective deep reinforcement learning for efficient workload orchestration in extreme edge computing. *IEEE Access*, 12:74558–74571, 2024.

[12] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint*, arXiv:1707.06347, 2017.

[13] A.M. Seid, G.O. Boateng, B. Mareri, G. Sun, and W. Jiang. Multi-agent drl for task offloading and resource allocation in multi-uav enabled iot edge network. *IEEE Transactions on Network and Service Management*, 18(4):4531–4547, 2021.

[14] S. Shen, Y. Han, X. Wang, S. Wang, and V.C. Leung. Collaborative learning-based scheduling for kubernetes-oriented edge-cloud network. *IEEE/ACM Transactions on Networking*, 31(6):2950–2964, 2023.

[15] M. Su, G. Wang, and K.K.R. Choo. Prediction-based resource deployment and task scheduling in edge-cloud collaborative computing. *Wireless Communications and Mobile Computing*, 2022(1):2568503, 2022.

[16] J. Wang, M. Zhang, Q. Yin, L. Yin, and Y. Peng. Multi-agent reinforcement learning for task offloading with hybrid decision space in multi-access edge computing. *Ad Hoc Networks*, 166:103671, 2025.

[17] X. Wang, K. Zhao, and B. Qin. Optimization of task-scheduling strategy in edge kubernetes clusters based on deep reinforcement learning. *Mathematics*, 11(20):4269, 2023.

[18] Z. Wang, M. Goudarzi, M. Gong, and R. Buyya. Deep reinforcement learning-based scheduling for optimizing system load and response time in edge and fog computing environments. *Future Generation Computer Systems*, 152:55–69, 2024.

[19] X. Yang and J.A. Esquivel. Lstm network-based adaptation approach for dynamic integration in intelligent end-edge-cloud systems. *Tsinghua Science and Technology*, 29(4):1219–1231, 2024.

[20] S.A. Zakaryia, M. Meaad, T. Nabil, and M.K. Hussein. Task offloading and resource allocation for multi-uav asset edge computing with multi-agent deep reinforcement learning. *Computing*, 107(5):1–31, 2025.

[21] L. Zhang, F. Xu, R. Wang, and X. Huang. Kubernetes edge server resource prediction model integrating vmd and lstm, 2023. Available at SSRN 4573647.

[22] T. Zheng, J. Wan, J. Zhang, and C. Jiang. Deep reinforcement learning-based workload scheduling for edge computing. *Journal of Cloud Computing*, 11(1):3, 2022.