Smart Orchestration: Novel Machine Learning-Driven Service Management at the Edge

Thiago F. Ribeiro, Talles Magalhães, Glauco Goncalves

[1] Federal University of Pará (UFPA), Belém, PA, Brazil

thiago.ribeiro@ig.ufpa.br,

talles.magalhaes@itec.ufpa.br,

glauco.goncalves@lasse.ufpa.br

1. Introduction

The rapid growth of edge computing has revolutionized the way data are processed and managed, particularly in latency-sensitive applications such as Internet of Things (IoT) systems, autonomous vehicles, and real-time analytics. Edge servers, positioned closer to end-users, are pivotal in handling decentralized data and reducing communication latency. However, the management of dynamic and heterogeneous service workloads poses significant challenges, such as resource allocation, fault tolerance, and maintaining service quality under varying conditions [Rahamathulla and Ramaiah 2024]. Innovative approaches leveraging machine learning have shown promise in addressing these complexities.

Traditional resource management techniques in edge environments often fail to adapt to their highly dynamic nature, resulting in issues such as resource underutilization, service bottlenecks, and increased latency, which undermine overall efficiency [Aslanpour et al. 2020]. To address these challenges, recent advancements in machine learning have been explored for edge computing applications. While single-model approaches can optimize specific tasks, their effectiveness is frequently constrained by model biases and limited generalization capabilities. To overcome these limitations, ensemble learning—which integrates multiple models to enhance prediction accuracy and robustness—has emerged as a promising strategy for effective service management on edge servers [Bai et al. 2022, Sikdokur et al. 2023, Petri et al. 2018]. Additionally, reinforcement learning techniques are being investigated to enable adaptive and intelligent resource allocation [Rjoub et al. 2021], offering comprehensive solutions to the inefficiencies inherent in traditional resource management methods. Ensemble and reinforcement learning provide robust frameworks for optimizing resource utilization and improving the performance of edge computing infrastructures.

2. Related Works

The growing need for energy-efficient service migration in cloud and edge computing has inspired several AI-driven approaches. [Rjoub et al. 2021] proposed a hybrid DRL-LSTM model to optimize task scheduling in large-scale cloud systems. Their approach demonstrated significant reductions in CPU and memory usage compared to traditional methods like shortest job first (SJF) and improved particle swarm optimization (PSO).

The combination of reinforcement learning and sequence-based models proved effective in managing resources dynamically while minimizing delays.

[Kumar et al. 2023] focused on edge computing with the development of EdgeAISim, a Python-based toolkit for simulating AI-based resource management. By leveraging models such as Deep Q-Networks and Actor-Critic Networks, EdgeAISim reduced power consumption and optimized task migration, promoting sustainable computing practices. The toolkit's modularity allows researchers to test and refine policies in diverse edge environments.

In healthcare, [Shinoo et al. 2024] introduced an Ensemble Extreme Learning Machine (EN-ELM)-based Clinical Decision Support System (CDSS). This system employed edge computing to minimize latency while addressing outliers and class imbalances in medical datasets. The integration of adaptive synthetic oversampling and isolation forests enhanced prediction accuracy while maintaining energy efficiency. The CDSS highlights the potential of edge computing for real-time, resource-efficient applications in critical domains.

These works underscore the transformative potential of AI and edge computing in addressing energy consumption challenges during service migration. Building on this foundation, the proposed ensemble machine learning framework extends current advancements by integrating bagging, boosting, and stacking techniques to dynamically manage workloads and predict service demands in edge server environments. Unlike traditional approaches, this framework emphasizes optimizing resource allocation and balancing loads while ensuring system reliability. By validating its scalability and adaptability through extensive experimentation, this study contributes to the ongoing evolution of energy-efficient edge computing systems, aligning with the latest research and pushing the boundaries of smart service management.

3. Theoretical Background and State of the Art

3.1. Edge Computing Overview

Edge computing has rapidly emerged as a pivotal technology for applications that demand low latency and real-time processing, such as autonomous vehicles, Industrial Internet of Things (IIoT), and smart city infrastructures. By situating computational resources closer to end-users, edge servers significantly reduce data transmission latency and alleviate the processing burden on centralized cloud systems. This proximity not only enhances response times but also improves data privacy and reduces bandwidth usage. Despite these advantages, edge computing faces several persistent challenges, including scalability, heterogeneity, and reliability [Ros et al. 2024]. Scalability issues arise from the need to manage an ever-increasing number of edge devices and services, while heterogeneity pertains to the diverse hardware and software configurations across different edge nodes. Reliability remains a concern due to the distributed nature of edge environments, which can be more susceptible to failures and disruptions compared to centralized systems. To address these challenges, contemporary approaches are increasingly integrating intelligent resource management techniques, such as dynamic load balancing and predictive maintenance, into the architecture of edge servers. These advancements aim to enhance the efficiency, robustness, and adaptability of edge computing infrastructures, ensuring they can meet the demanding requirements of modern applications.

3.2. Service Management in Edge Servers

Effective service management within edge servers is crucial for maintaining high levels of system responsiveness and ensuring consistent service quality. Traditional service management techniques often rely on heuristic or rule-based methods, which, while straightforward to implement, lack the flexibility to adapt dynamically to fluctuating workloads and changing environmental conditions. These methods can lead to suboptimal resource utilization, increased latency, and reduced throughput, particularly in scenarios with highly variable demand. In contrast, machine learning-based approaches, encompassing both supervised and unsupervised learning paradigms, have demonstrated significant potential in enhancing service management. These approaches enable dynamic task offloading, efficient resource allocation, and robust fault tolerance by learning from historical data and adapting to real-time conditions [Lim and Joe 2024]. For instance, supervised learning models can predict future workloads based on past trends, allowing for proactive resource provisioning, while unsupervised learning techniques can identify anomalies and optimize resource distribution without explicit labels. By leveraging these advanced models, edge computing systems can achieve optimized resource usage, maintain low latency, and ensure high throughput, which are essential for critical applications such as real-time monitoring, autonomous navigation, and responsive IoT services.

3.3. Machine Learning in Edge Computing

Machine learning (ML) has become increasingly integral to the advancement of edge computing, providing sophisticated tools for managing and optimizing distributed resources. Among the various ML techniques, deep learning models, particularly reinforcement learning (RL), have been extensively utilized for tasks such as predictive workload management, resource optimization, and anomaly detection within edge environments. RL algorithms enable systems to learn optimal policies through interactions with their environment, making them well-suited for dynamic and uncertain settings typical of edge computing. Recent developments in multi-agent reinforcement learning (MARL) have further enhanced these capabilities by facilitating collaborative decision-making processes across multiple edge servers [Koo and Park 2024]. MARL allows different agents to share information and coordinate actions, leading to more efficient and scalable resource management strategies. Despite these advancements, single-model approaches often encounter limitations such as overfitting to specific scenarios and inherent biases that can degrade performance in diverse and evolving environments. To mitigate these issues, there has been a noticeable shift towards ensemble techniques, which combine multiple models to improve overall reliability and accuracy. Ensemble learning leverages the strengths of various models, reducing the likelihood of overfitting and enhancing the system's ability to generalize across different tasks and conditions, thereby providing more robust and resilient solutions for edge computing service management.

3.4. Ensemble Learning for Service Management

Ensemble learning represents a powerful strategy in machine learning, where multiple models are combined to achieve superior predictive accuracy and robustness compared to individual models. In the context of edge computing, ensemble techniques such as bagging, boosting, and stacking are particularly advantageous due to the complex and heterogeneous nature of edge environments. Bagging (Bootstrap Aggregating) improves model

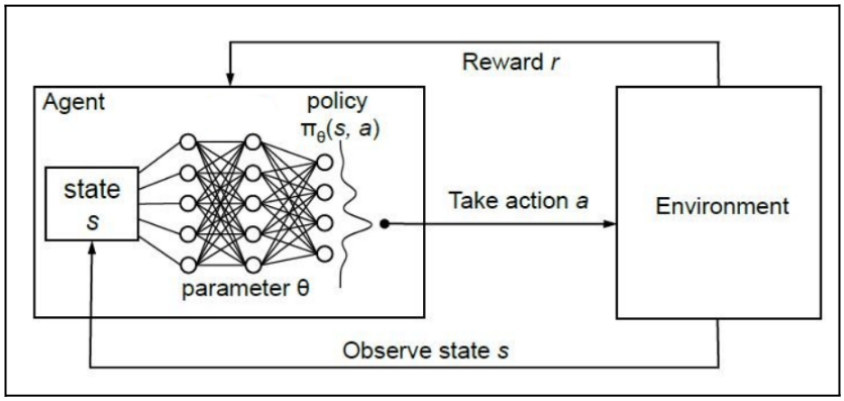

Figure 1. DRL Agent overview.

stability and accuracy by training multiple instances of a model on different subsets of the data and averaging their predictions. Boosting, on the other hand, focuses on sequentially training models by emphasizing previously misclassified instances, thereby enhancing the overall performance. Stacking involves training a meta-model to integrate the predictions of several base models, capturing diverse patterns and improving generalization. A recent study exemplifies the potential of ensemble models in optimizing resource allocation across distributed edge servers. Their research demonstrated that ensemble methods could effectively address key challenges such as fault tolerance, scalability, and adaptability by leveraging the combined strengths of multiple models [Lim and Joe 2024]. By integrating these ensemble techniques into edge service management frameworks, researchers aim to develop more resilient and efficient systems capable of handling the dynamic and unpredictable demands of modern edge computing applications. This integration not only enhances the accuracy of resource predictions but also ensures more stable and reliable service delivery, making ensemble learning a critical component in the evolution of intelligent edge computing infrastructures.

3.5. Reinforcement Learning (RL) and Proximal Policy Optimization (PPO) Overview

Reinforcement Learning (RL) has established itself as a robust and versatile framework for addressing sequential decision-making problems, where an agent learns to make optimal decisions through interactions with its environment. In RL, the agent receives feedback in the form of rewards or penalties based on its actions, enabling it to learn policies that maximize cumulative rewards over time. Among the diverse array of RL algorithms, Proximal Policy Optimization (PPO) has gained prominence due to its balance of efficiency and stability. PPO is a policy-gradient method that iteratively updates the agent's policy by optimizing a surrogate objective function, which ensures that the policy updates remain within a trust region to prevent drastic changes that could destabilize learning [Schulman et al. 2017]. The simulations steps is illustrated in figure 1.

Energy Per Service Migration in Edge Computing

Service migration is a fundamental aspect of edge computing, essential for maintaining uninterrupted service continuity, optimizing resource utilization, and adhering to stringent latency requirements. However, each migration event carries inherent energy costs, which

can significantly impact the overall efficiency and sustainability of edge systems. As edge computing underpins real-time applications such as augmented reality, Internet of Things (IoT) services, and autonomous systems, it is imperative to understand and mitigate the energy consumption associated with service migrations to ensure sustainable operation.

Recent research has provided valuable insights into the energy dynamics of service migration. For instance, [Ali et al. 2024] identified that the energy costs of service migration stem from several factors, including data transfer between nodes, the computational overhead required to reinitialize services, and the synchronization of service states across different edge servers. To address these challenges, they proposed energy-aware resource allocation mechanisms that achieved energy consumption reductions in the range of 15-25%. Similarly, [Hongchang et al. 2024] focused on developing intelligent algorithms aimed at minimizing migration energy costs. By leveraging deep reinforcement learning, their approach was able to identify optimal migration points and select target servers effectively, resulting in energy savings of up to 30% in dynamic environments. Additionally, Wirth et al. [Wirth et al. 2024] explored the additional energy costs incurred during service migrations due to system instabilities or node failures. They introduced probabilistic models to assess the energy trade-offs involved in migrating digital twins across edge nodes, taking into account uncertainties related to user mobility and network reliability.

The energy consumption associated with each migration event is influenced by various factors, including the chosen migration strategy, the underlying network topology, and the specific workload characteristics of the edge environment. Advanced techniques such as machine learning, fine-grained resource monitoring, and predictive analytics present promising avenues for reducing the energy footprint of edge computing systems. By integrating these technologies, it is possible to develop more efficient migration strategies that not only minimize energy consumption but also enhance the overall performance and reliability of edge services. Consequently, addressing the energy challenges of service migration is critical for the sustainable and scalable deployment of edge computing infrastructures, particularly as the demand for real-time and resource-intensive applications continues to grow.

4. Proposed Frameworks

4.1. Q-learning Ensemble Framework

The proposed framework introduces a multi-agent reinforcement learning system leveraging Deep Q-Network (DQN) agents to optimize decision-making in an edge computing environment. Each agent contributes to a vote on the model to assess the necessity of the change of a service from an edge server to edge server, enabling a distributed and scalable approach to managing computational resources or task offloading.

The key innovation lies in the deployment of multiple DQN agents, where each agent operates based on the state and action spaces defined by the characteristics of the edge servers. Specifically, the framework scales the state space as the number of agents, capturing multi-dimensional metrics such as current load and energy consumption. By encoding these features into the state space, the DQN agents gain a holistic understanding of the system dynamics, enabling efficient and adaptive decision-making.

4.1.1. State and Action Spaces

The EdgeServer class and its `all()` method form the foundation for defining the agents' state and action spaces. Each edge server contributes a set of metrics to the state space, resulting in a state representation of size $4 \times N$, where N is the number of edge servers. These metrics include:

- **Current load:** The utilization of CPU, memory and disk resources on each edge server.
- **Energy consumption:** The power usage incurred during service migration.

The action space consists of decisions for resource allocation or task offloading, where each action corresponds to assigning tasks to specific edge servers.

However, scaling the state space uniformly across all agents may introduce computational overhead, particularly in large-scale edge environments. As the number of edge servers increases, the dimensionality of the state space grows proportionally, potentially leading to slower convergence and higher training complexity for the DQN agents.

4.1.2. Multi-Agent Design and Scalability

The proposed multi-agent framework reflects the distributed nature of edge networks, where each DQN agent learns independently while sharing the global objective of system optimization. This decentralized approach enhances the scalability of the framework, allowing it to accommodate an increasing number of edge servers without centralized bottlenecks.

Nonetheless, the uniform design of agents does not account for the inherent heterogeneity of real-world edge servers, which often vary in computational power, energy efficiency, and network capabilities. Future enhancements could introduce agent specialization, where agents are tailored to specific server characteristics, improving overall adaptability and performance in heterogeneous environments.

4.2. DRL-based Framework Architecture

The proposed framework builds upon the EdgeSimPy simulator, customized for modeling service migrations and resource management in edge computing. Wrapped in Python using the Gymnasium interface, the framework integrates seamlessly with RL algorithms, enabling efficient training and evaluation. The main components of the framework are detailed below:

4.2.1. Environment Implementation with EdgeSimPy

EdgeSimPy serves as the core simulation environment, offering comprehensive modeling capabilities for service allocation and migration in edge systems. It dynamically adjusts resource utilization and computes energy consumption, providing a realistic environment for RL agents.

4.2.2. Autonomous Process Flow

The simulation process is entirely automated. At the start, the environment initializes servers and services based on a predefined dataset. Each step involves evaluating the current state, executing agent-selected actions, and updating server capacities and energy consumption. The agent decides whether to migrate services to different servers or retain them, considering constraints such as CPU, memory, and disk utilization. The environment logs power consumption and service allocations at every step, enabling detailed performance analysis.

4.2.3. Resource Allocation by RL Agent

The RL agent interacts with the EdgeSimPy environment to optimize service placement and migration decisions.

4.2.4. Python Wrapper with Gymnasium Interface

The environment is wrapped in the Gymnasium interface to facilitate interaction with RL libraries. The action space is defined as a multi-discrete space, allowing the agent to decide on the destination server for each service or opt not to migrate. Observations include server utilization metrics, allocation matrices, and exploration scores generated by a multi-armed bandit (MAB) algorithm.

4.2.5. Multi-Armed Bandit for Exploration

To guide initial exploration, a MAB algorithm is incorporated. It calculates Upper Confidence Bound (UCB) scores for each server, balancing the exploration of underutilized servers with the exploitation of high-performing ones. This ensures efficient exploration during the early training phases, improving overall agent performance.

By integrating PPO and EdgeSimPy, this framework offers a robust and scalable solution for optimizing energy efficiency in edge computing. It sets the foundation for further advancements in dynamic resource management and sustainable system design.

5. Experimental Setup

5.1. Testbed Configuration

The experimental setup for this study builds upon the server configuration detailed in the paper EdgeAISim: A toolkit for simulation and modelling of AI models in edge computing environments [Kumar et al. 2023]. The EdgeAISim toolkit provides a robust environment for simulating edge computing systems and evaluating the performance of machine learning models in resource-constrained scenarios. The specifications for the servers hardware simulated in our experiments are as follows:

Server	CPU	Memory	Disk
Edgeserver1	8	16384	131072
Edgeserver2	8	16384	131072
Edgeserver3	8	8192	131072
Edgeserver4	8	8192	131072
Edgeserver5	12	16384	131072
Edgeserver6	12	16384	131072

Table 1. Servers Specifications

This hardware is representative of a modern edge server, enabling us to evaluate the proposed ensemble learning framework in realistic edge computing scenarios.

5.2. Workload and Dataset Simulation

The experiments utilize workloads and datasets designed to reflect real-world edge computing use cases. Simulated workloads include IoT-generated data streams, video analytics tasks, and sensor data processing for industrial IoT applications. The datasets for training and testing are derived from publicly available benchmarks such as:

- The OpenEdge dataset for IoT sensor data.
- Custom workloads generated using the EdgeAISim toolkit to emulate heterogeneous tasks.

The workloads are configured to test the system's adaptability under varying conditions, including bursty traffic and resource contention.

5.3. Baseline Models for Comparison

To validate the performance of the proposed ensemble learning framework, its results are compared against baseline models introduced in EdgeAISim [Kumar et al. 2023]. These baseline models include:

- **Single Model Approaches:** Worst Fit (state-wise) and Multi-Armed Bandit.
- **Reinforcement Learning Models:** Deep Q-Networks (DQN) and Deep Q-networks with Graphical Neural Networks (DeepQ+GNN) for dynamic service distribution.
- **Deep Reinforcement Learning Models:** PPO.

The metrics used for comparison include Number of service migrations, total simulation time, average power on migration, power por migration..

5.4. Experimental Procedure

The experimental workflow comprises the following steps:

1. **Model Training:** Train all models, including baseline and ensemble frameworks, using the training datasets. Hyperparameter tuning is performed using grid search for each model.
2. **Deployment:** Deploy models on the simulated edge environment using the EdgeAISim toolkit.
3. **Performance Measurement:** Measure the models' performance using standardized metrics, Number of service migrations, total simulation time, average power on migration, power por migration.

5.5. Evaluation Metrics

The evaluation focuses on the following metrics to ensure comprehensive performance assessment:

- **Number of Service Migrations:** Total count of service migrations occurring during the simulation.
- **Total Simulation Time:** The overall time taken to complete the simulation process.
- **Average Power on Migration:** The average power consumption measured during service migration events.
- **Power per Migration:** The power consumption attributed to each individual service migration.

6. Results and Discussion

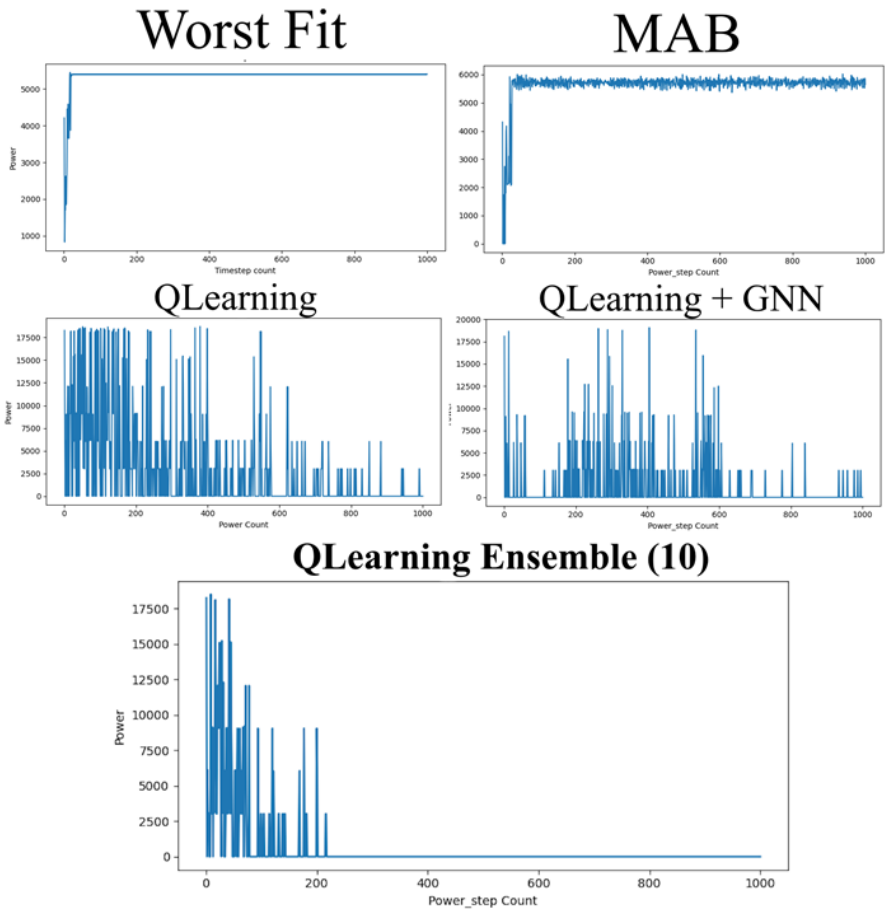

Figure 2. Power Consumption Trends for Different Models

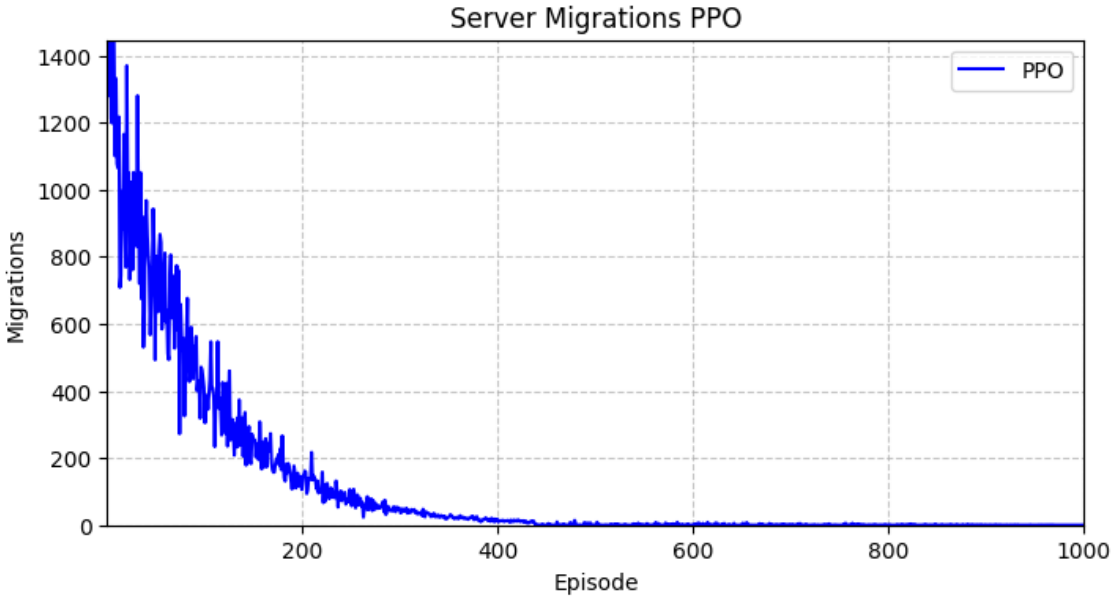

Figure 3. Server Migrations PPO Evaluation

6.1. Performance Comparison of Models

The performance evaluation compares our proposed ensemble Q-Learning framework (QE) to baseline models (Worst Fit, MAB) and existing models from EdgeAISim [Kumar et al. 2023], including Q-Learning and GNN-Q-Learning. The results are assessed based on service migrations, total simulation time, and average power consumption per migration.

Table 2 illustrates the quantitative results. The baseline Worst Fit and EdgeAISim's MAB models exhibit high service migrations (5961 and 5909, respectively), which correlate with increased average power consumption (5366 W and 5630 W). The Q-Learning model significantly improves over these baselines with only 933 migrations and 2865 W average power consumption. However, GNN-Q-Learning outperforms all other models in terms of migrations (100) and simulation time (6764 ms), albeit at a higher power consumption of 1228 W.

Our proposed QE (15) achieves a balanced performance, reducing average power consumption to 476 W while maintaining a low number of migrations (157). This result highlights the ensemble model's efficiency in managing services while conserving power resources.

6.2. Effect of Ensemble Size on Performance

The impact of ensemble size on the number of service migrations, power consumption, and simulation time is presented in Figure 4. As the number of ensemble agents increases, the number of migrations decreases significantly from 980 (1 agent) to 19 (200 agents). Average power consumption also decreases initially, stabilizing around 54 W for 50+ agents. However, the total simulation time increases linearly, reaching 115 seconds for 200 agents.

Table 2. Performance Comparison of Models. SM = Server Migrations, TST: Total Simulation Time, AP: Average Power, APM: Average Power per Miration

Works	Model	SM	TST (ms)	AP (W)	APM (W)
Baseline	Worst Fit	5961	5231	5366	0.90
EdgeAISim	MAB	5909	20824	5630	0.95
EdgeAISim	Q-Learning	933	5079	2865	3.07
EdgeAISim	**GNN-Q-Learning**	**100**	**6764**	**1228**	**12.28**
Ensemble Q-Learning(Ours)	**QE (15)**	**157**	**6682**	**476**	**3.03**
Reinforcement learning (Ours)	**PPO**	**23**	**2687**	**528**	**22.9**

This trade-off highlights that increasing the ensemble size reduces service disruptions and improves energy efficiency at the cost of higher computational overhead. For practical deployment, an ensemble size of 15 agents offers the optimal balance between power efficiency (476 W) and computational time (6.6 seconds).

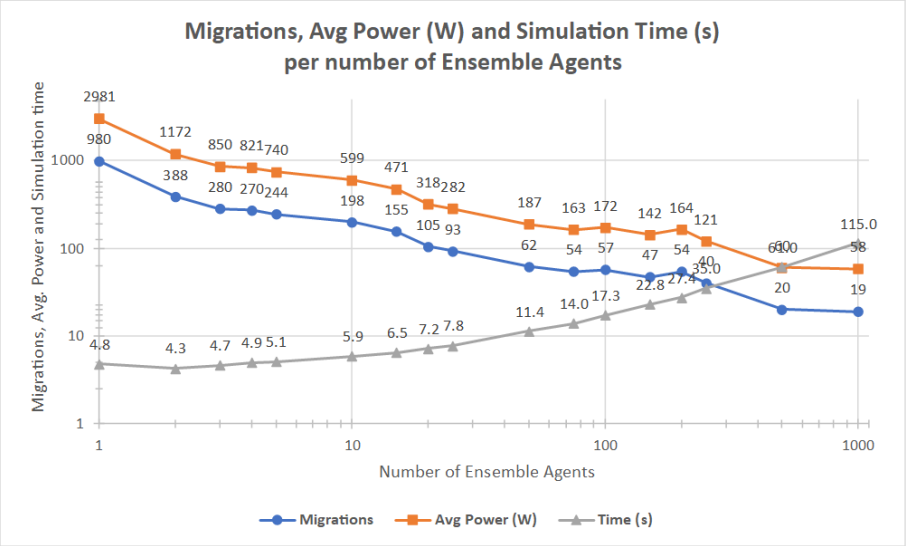

Figure 4. Impact of Ensemble Agents on Migrations, Power Consumption, and Simulation Time

6.3. PPO Performance

The graph in figure 2 illustrates the number of server migrations per training episode for the Proximal Policy Optimization method. It is evident from the graph that the number of migrations is initially high, with significant fluctuations during the early episodes of training. This reflects the exploratory behavior of the PPO algorithm as it attempts to learn optimal migration policies in a dynamic environment.

As training progresses, the number of migrations decreases consistently, indicating that the PPO model is gradually improving its decision-making capabilities and learning to minimize unnecessary server migrations. By approximately episode 400, the migration count stabilizes at a significantly lower level, suggesting that the model has converged to an efficient migration policy.

The reduction in migrations over time highlights the effectiveness of the PPO algorithm in optimizing resource allocation and minimizing migration-related overhead.

The residual fluctuations observed after convergence are a result of the inherent variability in the training environment and the stochastic nature of the reinforcement learning process. These fluctuations are minimal, demonstrating that the learned policy is robust and capable of maintaining stability under varying conditions.

6.4. Power Consumption Trends

Figure 2 compares the power consumption trends for all evaluated models. The baseline models (Worst Fit and MAB) exhibit consistently high power usage, while the Q-Learning models demonstrate significant power savings. GNN-Q-Learning, despite its lower migration count, incurs higher power consumption due to computational overhead.

Our QE (15) model achieves the lowest average power consumption while maintaining competitive performance in reducing migrations. The fluctuations observed in Q-Learning and GNN-Q-Learning power usage highlight their inability to stabilize resource management compared to the ensemble-based approach.

6.5. Discussion

The experimental results demonstrate the effectiveness of the proposed ensemble Q-Learning framework for service management in edge servers. The key findings are:

- Our QE (15) model achieves a significant reduction in power consumption (476 W) compared to GNN-Q-Learning (1228 W) while maintaining a low number of migrations (157).
- Increasing ensemble size improves performance (fewer migrations, lower power usage) but introduces higher simulation time overhead.
- The results validate the efficiency of ensemble learning in stabilizing resource allocation and optimizing power usage in dynamic edge environments.

These findings underscore the potential of ensemble machine learning to balance power efficiency, service reliability, and computational cost in edge computing environments.

References

Ali, B., Golec, M., Gill, S., Wu, H., Cuadrado, F., and Uhlig, S. (2024). Edgebus: Co-simulation based resource management for heterogeneous mobile edge computing environments. *Internet of Things*.

Aslanpour, M. S. et al. (2020). Performance evaluation metrics for cloud, fog and edge computing: A review, taxonomy, benchmarks and standards for future research. *Internet of Things*, 12:100273.

Bai, Y., Chen, L., Abdel-Mottaleb, M., and Xu, J. (2022). Automated ensemble for deep learning inference on edge computing platforms. *IEEE Internet of Things Journal*, 9(6):4202–4213.

Hongchang, K., Hui, W., and Hongbin, S. (2024). Deep reinforcement learning-based task offloading and service migrating policies in service caching-assisted mobile edge computing. *IEEE Internet of Things Journal*.

Koo, M. and Park, J. (2024). Mec server status optimization framework for energy efficient mec systems by taking a deep-learning approach. *Future Internet*.

Kumar, A., Gupta, P., Singh, R., and Verma, S. (2023). Edgeaisim: A toolkit for simulation and modelling of ai models in edge computing environments. *Measurement: Sensors*, 24:100939.

Lim, D. and Joe, I. (2024). Maars: Multiagent actor–critic approach for resource allocation and network slicing in multiaccess edge computing. *Sensors*.

Petri, I., Zamani, A. R., Balouek-Thomert, D., Rana, O., Rezgui, Y., and Parashar, M. (2018). Ensemble-based network edge processing. In *2018 IEEE/ACM 11th International Conference on Utility and Cloud Computing (UCC)*, pages 133–142.

Rahamathulla, M. and Ramaiah, M. (2024). An intrusion attack classification using bio-inspired optimization technique and ensemble learning model for edge computing environments. *Multimedia Tools and Applications*.

Rjoub, G. et al. (2021). Deep and reinforcement learning for automated task scheduling in large-scale cloud computing systems. *Concurrency and Computation: Practice and Experience*, 33(23):e5919.

Ros, S., Kang, S., Song, I., Cha, G., Tam, P., and Kim, S. (2024). Priority/demand-based resource management with intelligent o-ran for energy-aware industrial internet of things. *Processes*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

Shinoo, A. C., Vincent, R., and Sengan, S. (2024). Edge computing-based ensemble learning model for healthcare decision systems. *Scientific Reports*, 14:26997.

Sikdokur, I., İnci M. Baytaş, and Yurdakul, A. (2023). Edgeconvens: Convolutional ensemble learning for edge intelligence.

Wirth, M., Ortiz, A., and Klein, A. (2024). Risk-aware bandits for digital twin placement in non-stationary mobile edge computing. In *IEEE International Conference on Communications*. IEEE.