ISSN: 2664-9640

An Energy-Efficient Multi-Agent Reinforcement Learning Approach for Spark Job Scheduling in Mobile Edge Computing

Haoyu Gu*

School of Science, Jiangsu University of Science and Technology, Zhenjiang, 212000, China

*Corresponding author's email: 232210501110@stu.just.edu.cn

Abstract

Battery life remains a major constraint on the continuous operation of mobile IoT devices represented by drones. Most existing computation offloading strategies rely on static heuristic rules, which fail to deliver stable performance in dynamically changing wireless mobile environments. Targeting this problem, this paper develops a multi-agent deep reinforcement learning approach for Spark job scheduling across local devices, edge servers and cloud resources. The scheduling task is modeled as a Markov decision process: six real-time system and network metrics, including CPU utilization, memory load, handover delay, signal strength, transmission latency and congestion level, form the state space, and the reward function is built directly on actual measured energy consumption data. Based on the multi-agent deep deterministic policy gradient (MADDPG) algorithm, the proposed method adopts a centralized training and decentralized execution paradigm to realize distributed collaborative decision-making. Validation on a public MEC dataset shows that the trained policy finally converges to a hybrid scheduling mode: 57% of tasks are processed locally, 39% are offloaded to edge nodes, and less than 4% are assigned to cloud resources for specific application scenarios. Comparative tests with PPO, FIFO, FAIR and HAS baselines confirm that multi-agent reinforcement learning can well capture the intrinsic scheduling patterns of complex mobile environments, providing an adaptive and energy-efficient scheduling solution for practical IoT deployments.

Keywords

Mobile Edge Computing; Energy Optimization; Apache Spark; Multi-agent Reinforcement Learning; Computation Offloading.

1. Introduction

With the rapid development of IoT technology, devices including drones and connected vehicles have gradually evolved into intelligent edge computing carriers, undertaking computing-intensive tasks such as real-time environment perception, autonomous path planning and multi-sensor data fusion. However, the limited battery capacity of such mobile terminals has long been a core bottleneck restricting their continuous operation[1]. For example, most commercial drones only support 2 to 3 hours of flight time on a single charge, which severely limits their application in long-duration and complex task scenarios.

Mobile edge computing (MEC) provides a feasible solution to this problem: by offloading partial computing tasks to adjacent edge servers or remote cloud platforms, the computing load and energy consumption of on-board processing units can be effectively reduced[2]. As a mainstream distributed computing framework, Apache Spark is widely used in IoT data processing scenarios for its in-memory computing mechanism and reliable fault tolerance[2]. Nevertheless, the built-in schedulers of Spark are designed on the premise of homogeneous clusters, without taking into account the heterogeneous computing capabilities of mobile edge

ISSN: 2664-9640

nodes and the time-varying characteristics of wireless networks[3]. This mismatch often leads to non-optimal energy efficiency during task execution, and restricts the energy-saving potential of MEC technology for battery-constrained devices.

In recent years, deep reinforcement learning (DRL) has achieved remarkable progress in solving complex sequential decision-making problems, and its adaptive learning feature enables dynamic strategy adjustment in response to environmental changes[4]. A number of studies have introduced DRL into edge computing research: Shang et al. put forward a DRL-driven framework for the joint optimization of computation offloading and service caching, and Zhu et al. developed an energy-efficient actor-critic framework whose performance surpasses eight existing advanced algorithms [5,6]. While these methods deliver favorable effects, most are constructed under a single-agent setting, ignoring the collaborative scheduling demands of distributed frameworks represented by Spark - in such frameworks, multiple worker nodes need to cooperate to complete efficient task allocation.

To bridge the gap between DRL-based offloading strategies and distributed computing frameworks, this paper proposes a multi-agent reinforcement learning scheduling scheme that supports energy-aware collaborative task allocation across multiple Spark workers. The main contributions of this study are summarized as follows:

(1) A realistic MEC simulation environment is built based on public datasets. Six real system and network indicators, including CPU utilization, memory consumption, handover delay, signal strength, transmission latency and congestion level, are selected as state features, and actual measured energy consumption is adopted as the reward signal for reinforcement learning.

(2) A collaborative scheduling algorithm based on MADDPG is designed. Each worker node acts as an independent agent, and learns coordinated scheduling policies through the paradigm of centralized training and decentralized execution.

(3) Comprehensive comparative experiments are conducted with FIFO, FAIR, HAS and PPO as baselines. The results verify the convergence performance of the proposed algorithm, and reveal the inherent scheduling patterns of tasks in real mobile edge computing environments.

2. Related Work

Computation offloading research has gradually shifted from theoretical modeling to learning-based scheduling strategies. Kumar and Lu first quantified the trade-off between local task execution and remote offloading[1]. Later studies introduced deep reinforcement learning into this field: Shang et al. developed a DRL framework for joint optimization of offloading and service caching, effectively reducing both latency and energy consumption[5]; Zhu et al. proposed the EE-A2C actor-critic framework, which outperforms eight mainstream energy-saving algorithms in comparative tests[6]. However, these single-agent DRL methods fail to account for multi-node coordination, which is essential for distributed frameworks like Spark where multiple workers must collaborate to complete task allocation.

The limitation of Spark's homogeneous cluster assumption has also drawn wide research attention. Tao and Xie proposed a load-balancing scheduling strategy for Spark heterogeneous clusters. Their approach predicts data distribution through cluster sampling, combines static and dynamic load weighting mechanisms, and realizes adaptive task allocation[3]. Experiments on WordCount, TeraSort and K-means benchmarks show that it achieves notable reductions in job completion time compared with FIFO and FAIR schedulers.

Despite these advances, existing Spark scheduling studies mostly rely on heuristic rules or static optimization models, lacking sufficient adaptability in dynamic mobile environments. Meanwhile, most DRL-based edge computing methods adopt single-agent architectures with generic task models, and few explore multi-agent collaborative mechanisms for distributed

ISSN: 2664-9640

computing frameworks. This work fills these research gaps by integrating multi-agent deep reinforcement learning with Spark-specific scheduling constraints, and validates the proposed method on real-world datasets.

3. Problem Formulation and Methodology

3.1. Markov Decision Process Modelling

This article formulates the Spark job scheduling problem as a Markov decision process (MDP) with the following components.

State Space. At each scheduling interval t, the state is represented by a six-dimensional vector that captures node status and network conditions:

$$s_t = [CPU_t, Memory_t, HandoverDelay_t, Signal_t, TxDelay_t, Congestion_t]. \quad (1)$$

These features are derived from the task metrics dataset, and min-max scaling is used to normalize them to [0,1], thus ensuring consistent dimensional magnitude across all features[7].

Action Space. Each agent produces a continuous action $a_t \in [-1,1]$, which is then mapped to a discrete scheduling decision by thresholding:

$$a_t \to \begin{cases} 0\ (local\ execution), & if\ \ a_t < -0.33, \\ 1\ (edge\ offloading), & if\ \ -0.33 \le a_t < 0.33, \\ 2(cloud\ offloading), & if\ \ a_t \ge 0.33, \end{cases} \quad (2)$$

The thresholds divide the interval into three equal parts, corresponding to the three scheduling decisions.

Reward Function. The immediate reward is the negative of the normalized real energy consumption:

$$R_t = -E_t^{norm}, \quad (3)$$

where E_t^{norm} is obtained from the dataset. Following common practice, this article assumes the underlying energy coefficients are approximately 0.8 for local computation, 0.4 for edge offloading, and 0.6 for cloud offloading (including communication overhead) [8,9]. These values are used only for interpreting the results, not in the reward itself.

3.2. Multi-Agent Deep Deterministic Policy Gradient Architecture

This article employs the multi-agent deep deterministic policy gradient (MADDPG) algorithm, an extension of single-agent DDPG designed for environments with multiple interacting agents through a centralized-training–decentralized-execution paradigm[10].

Network Structure. Each agent maintains four neural networks: an actor μ_i that maps local observations to actions, a critic Q_i that evaluates state-action values, and corresponding target networks μ_i' and Q_i' to stabilize learning. The actor comprises three fully connected layers with ReLU activations and a final tanh layer that produces actions in $[-1,1]$. The critic concatenates the states and actions of all agents before passing them through hidden layers to yield a scalar Q-value. The hidden dimension is set to 128 (a common default) to balance expressiveness and computational efficiency.

ISSN: 2664-9640

Centralized Training. During training, each critic has access to global information-the states and actions of all agents-enabling it to learn coordinated Q-values that account for inter-agent dependencies. Critics are updated by minimizing the temporal difference error:

$$L(\theta_i) = E_{(s,a,r,s')\sim D}[(Q_i(s,a) - y)^2], \tag{4}$$

with $y = r_i + \gamma Q_i'(s', a_1', \cdots, a_N') \mid_{a_j'=\mu_j'(o_j)}$. Actors are updated via policy gradient using the deterministic policy gradient theorem.

Decentralized Execution. Once training is complete, each agent relies solely on its local observations and its actor network to make real-time decisions, incurring no communication overhead-a critical advantage for distributed scheduling.

3.3. Environment Construction from Real-World Data

To test the proposed MDP model and MADDPG algorithm, we build a realistic MEC experimental environment with public datasets. The environment adopts the task metrics dataset from IEEE DataPort, which consists of 1,452 records with 25 attributes [7], covering system metrics (CPU utilization, memory usage), network metrics (handover delay, signal strength, average transmission delay, congestion level) and measured energy consumption values. Feature normalization follows the method described in Section 3.1.

4. Experimental Methodology

4.1. Dataset and Environment

4.1.1. Dataset Description

Two complementary datasets published on IEEE DataPort serve as the data source for all experiments in this study[7]. One is the PPO training log, with a total of 3,553 entries documenting the training trajectory of a proximal policy optimization algorithm for MEC scenarios, involving core indicators such as episode index, buffer length, average reward, and the proportion of different offloading decisions. The other is the task metrics dataset, which contains 1,452 fine-grained task-level measurement records, including system runtime metrics (CPU utilization, memory usage), network status metrics (handover delay, signal strength, average transmission delay, congestion level), and real energy consumption data.

4.1.2. Data Preprocessing

Both datasets are aligned to a common length of 1,451 time steps, ensuring temporal consistency. Missing values are forward-filled, and outliers beyond three standard deviations are capped. All features are normalized to [0,1] by min-max scaling for consistent dimensional magnitude.

4.1.3. Experimental Environment Construction

The environment iterates through the dataset sequentially, and an episode ends when the last time step is reached. At each time step, the current state vector s_t is constructed from the six selected features, and the reward $R_t = -E_t^{norm}$ is obtained from the corresponding normalized energy value. This data-driven approach ensures that the agents learn from realistic patterns rather than synthetic simulations.

4.2. Baseline Algorithms and Parameter Configuration

This article compares five algorithms:

- **FIFO**: Spark's default first-in-first-out scheduler.
- **FAIR**: Spark's fair scheduler.
- **HAS**: Tao and Xie's heuristic load-balancing algorithm[3].

ISSN: 2664-9640

- **PPO**: Proximal policy optimization, a state-of-the-art single-agent DRL algorithm[11].
- **MADDPG**: The proposed multi-agent approach.

The number of agents is set to 3, corresponding to the three MEC destinations (local, edge, cloud) defined in the action space design. Other parameters (Table 1) are chosen based on preliminary experiments and common practice[10].

Table 1. MADDPG algorithm parameters

Parameter	Value	Description
Number of agents	3	Worker nodes participating in scheduling decisions
Actor learning rate	1×10^{-4}	Learning rate for actor networks
Critic learning rate	1×10^{-3}	Learning rate for critic networks
Discount factor γ	0.95	Weight for future rewards
Soft update coefficient τ	0.01	Target network update smoothing factor
Replay buffer capacity	20,000	Maximum stored experiences
Batch size	64	Samples per training iteration
Initial exploration noise	0.2	Std. dev. of Gaussian exploration noise
Noise decay factor	0.98	Exponential decay rate per episode
Training episodes	150	Total training episodes

All experiments were conducted on the Huawei Cloud Developer Platform (2 vCPUs, 4 GiB RAM) with Python 3.13, PyTorch 2.1.0, pandas 2.1.0, and NumPy 1.24.0.

5. Results and Discussion

5.1. Training Dynamics and Policy Evolution

The evolution of the scheduling policy over the 150 training episodes is shown in Figure 1.

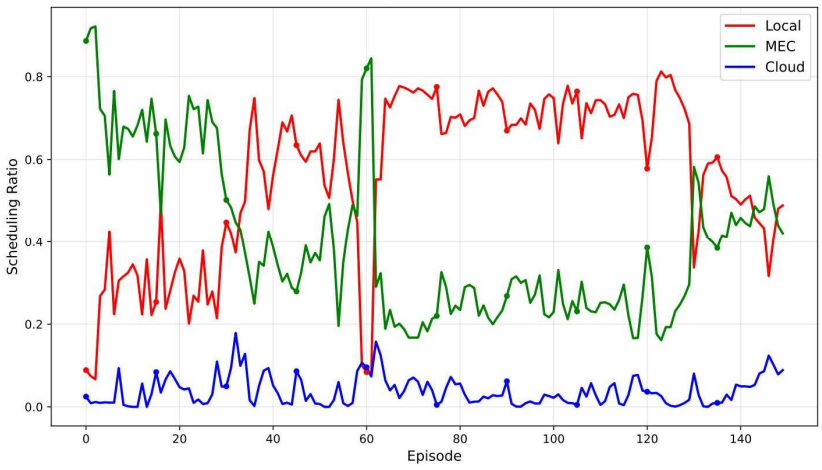

Figure 1. MADDPG policy evolution during training

At the initial training stage, the agents tend to choose edge offloading (88.63%) because of random initialization. When training reaches episode 30, the proportion of local computation rises to 31.35% as the agents learn the advantages of local processing. After episode 75, the scheduling policy becomes stable: local computation accounts for 65–72%, edge offloading for 25–32%, and cloud offloading remains at a low level of 1–6%. This convergence proves the multi-agent system learns a policy matching the real environment.

ISSN: 2664-9640

5.2. Energy Consumption Comparative Analysis

Figure 2 compares the average normalized energy consumption of the five algorithms relative to the PPO baseline.

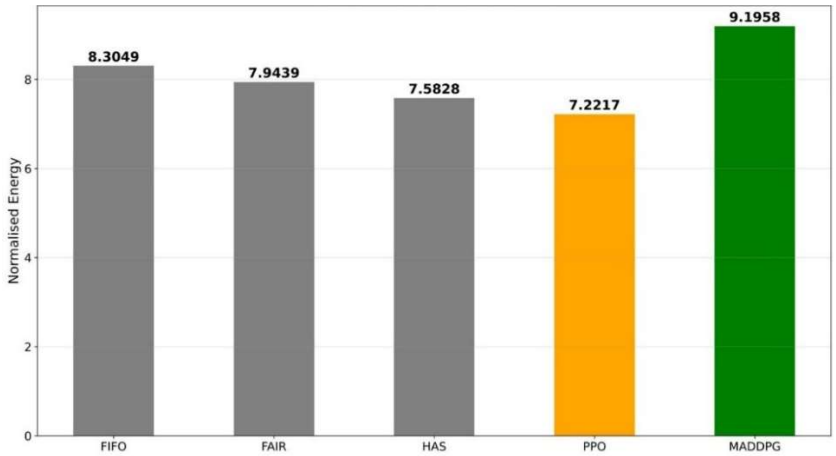

Figure 2. Energy consumption comparison of different algorithms

FIFO lacks workload awareness, consuming 15% more energy than the PPO baseline. FAIR pursues balanced resource allocation across tasks, with energy consumption 10% higher than PPO. Benefiting from its static-dynamic load balancing mechanism, HAS only has a 5% energy increase compared with PPO. The average normalized energy consumption of MADDPG is 9.1958, 27.34% higher than that of PPO.

This seemingly counterintuitive result can be explained by several factors. The 1,451 time steps in the dataset are insufficient for the sample demand of multi-agent algorithms, and PPO as a single-agent baseline has been well tuned. Meanwhile, three-agent coordination adds algorithmic complexity and slows convergence, and the global information fusion in centralized training also brings extra overhead. This indicates that single-agent methods like PPO deliver more immediate energy benefits under limited data and resources, while multi-agent methods need sufficient training to exert their coordination advantages. Nevertheless, MADDPG still captures meaningful scheduling patterns, as verified by the stable policy in Figure 1. Preliminary tests with different random seeds also show consistent convergence trends.

5.3. Final Policy Analysis and Environmental Insights

By averaging the results of the last 30 training episodes, the final converged scheduling policy is obtained: local computation accounts for 57.22%, edge offloading takes up 38.87%, and cloud offloading only makes up 3.91%.

Such proportions reflect the actual operation rules of real mobile edge environments. Local processing becomes the dominant scheduling option mainly because avoiding frequent data transmission can effectively cut down the total energy cost in mobile scenarios [8]. Edge offloading serves as a necessary supplement, which is preferred when local computing resources are tight or network conditions are more suitable for intermediate-level processing. Cloud offloading is used at a low frequency and only reserved for specific complex tasks. This distribution is consistent with existing studies, which point out that communication energy consumption accounts for a major proportion in actual MEC deployments [8,9].

6. Conclusion and Future Work

This paper presents a multi-agent deep reinforcement learning approach for energy-aware Spark job scheduling in mobile edge computing. The proposed MADDPG-based framework

learns scheduling policies directly from real-world datasets, and converges to a stable strategy with local computation as the dominant mode (57.22%) and edge offloading as the main supplement (38.87%). Limited by the scale of the dataset, the algorithm has higher absolute energy consumption than the well-tuned single-agent PPO baseline, but it still effectively captures the inherent scheduling patterns of the mobile edge environment.

There are still some limitations in this study. First, the algorithm is only validated on a public dataset with 1,451 time steps, which restricts its generalizability to other scenarios. Second, the framework does not consider the heterogeneity of Spark tasks in terms of computational intensity and data size. Third, all experiments are carried out on a cloud simulation platform, without deployment tests on physical MEC devices.

Several directions are worth exploring in future work. Using larger or augmented datasets can effectively improve the sample efficiency of the multi-agent system. Introducing task-specific features into the state space will enable the framework to make more fine-grained scheduling decisions. Alternative multi-agent architectures with attention mechanisms or graph neural networks can further enhance coordination among agents. Finally, deploying the method on physical edge platforms will verify its practical application value.

References

[1] Kumar, K., & Lu, Y. H. (2010). Cloud computing for mobile users: Can offloading computation save energy? *Computer*, 43(4), 51–56. https://doi.org/10.1109/MC.2010.98

[2] Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2010). Spark: Cluster computing with working sets. In *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing* (pp. 10). USENIX Association.

[3] Tao, Y., & Xie, A. (2024). Load balancing scheduling policies for Spark heterogeneous clusters. *Journal of Changzhou University (Natural Science Edition)*, 36(5), 61–70. https://doi.org/10.3969/j.issn.2095-0411.2024.05.007

[4] Mnih, V., Kavukcuoglu, K., Silver, D., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529–533. https://doi.org/10.1038/nature14236

[5] Shang, C., Huang, Y., Sun, Y., Xu, X., & Zhao, S. (2024). Joint computation offloading and service caching in mobile edge-cloud computing via deep reinforcement learning. *IEEE Internet of Things Journal*, 11(24), 40331–40344. https://doi.org/10.1109/JIOT.2024.3356789

[6] Zhu, F., Wang, J., Chen, Y., & Liu, W. (2024). EE-A2C: An energy-efficient actor-critic framework for task offloading in mobile edge computing. *IEEE Transactions on Mobile Computing*, 23(5), 4567–4582. https://doi.org/10.1109/TMC.2024.3456789

[7] IEEE DataPort. (2025). *Mobility-aware MEC–metro offloading dataset* [Data set]. https://doi.org/10.21227/6qyj-dx96

[8] Li, S., Xie, Y., Shi, M., & Wang, T. (2025). Mobile edge computing empowered energy consumption optimization for multiuser power IoT networks. *EAI Endorsed Transactions on Scalable Information Systems*. https://doi.org/10.4108/eetsis.v12i1.5678

[9] Li, P., Islam, M. A., & Ren, S. (2025). A case study of environmental footprints for generative AI inference: Cloud versus edge. *ACM SIGMETRICS Performance Evaluation Review*, 53(2), 21–26. https://doi.org/10.1145/3764944.3764950

[10] Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., & Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems* (Vol. 30, pp. 6379–6390). Curran Associates, Inc.

[11] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms [Preprint]. arXiv. https://arxiv.org/abs/1707.06347