# RSO-Optimized Federated Multi-Agent Actor-Critic RL for IoT Edge

Balla Shailesh Chowdary[1]
*Department of Information Technology*
*Vardhaman College of Engineering*
Hyderabad, India
ballashailesh@gmail.com

Karnam Poojitha[2]
*Department of Information Technology*
*Vardhaman College of Engineering*
Hyderabad, India
poojithak493@gmail.com

Bopparaju Meher Lakshmi Meghana[3]
*Department of Information Technology*
*Vardhaman College of Engineering*
Hyderabad, India
meghanabopparajumb@gmail.com

Ms. B. Swapna[4]
*Department of Information Technology*
*Vardhaman College of Engineering*
Hyderabad, India
swapna.dadigala@gmail.com
0000-0001-6224-2652

Ganesh B. Regulwar[5]
*Department of Information Technology*
*Vardhaman College of Engineering*
Hyderabad, India
ganeshregulwar@gmail.com
0000-0001-9320-7040

Dr K Ravindranath[6]
*Department of computer science*
*Koneru Lakshmaiah Education Foundation*
Hyderabad, India
ravindra$_{i}st@kluniversity.in$
0000-0001-6972-4857

***Abstract*—The explosive growth in Internet of Things (IoT) devices has resulted in severe resource allocation, energy consumption, and communication delay issues at the network's edge. This research proposes an optimized Federated Multi-Agent Actor-Critic Reinforcement Learning (FMAC-RL) model fortified with Red Fox Sparrow Optimization (RSO) to resolve the aforementioned challenges. The RSO algorithm is used to adaptively fine-tune hyperparameters in order to accelerate convergence, stabilize policies, and maximize overall system performance in diverse IoT scenarios. The suggested model of RSO-FMAC-RL facilitates IoT agents with decentralized learning in accordance with federated communication to ensure data confidentiality. Case study assessments on the simulation of IoT edge scenarios revealed appreciable improvement in performance with a Final Fitness of -487.67, Final Reward of -463.93, Latency of 0.00548 seconds, and an Energy Consumption of 0.4639 unit. The findings confirm the effective minimization of computational latency and the consumption of power with the optimization of collaborative task offload among the nodes at the edge. The proposed model of RSO-FMAC-RL creates an effective and power-aware learning model with scalable and privacy-enhancing IoT edge intelligence.**

**Keywords: Federated Learning, Multi-Agent Reinforcement Learning, Actor-Critic, Red Fox Sparrow Optimization, IoT Edge Computing, Energy Efficiency, Latency Reduction.**

## I. Introduction

Mobile Edge Computing (MEC) enables low-latency processing by bringing computation closer to end-users. Efficient task offloading is essential for IoT-driven systems, which must optimize latency, energy usage, and reliability [1]. Scalable approaches are necessary for complex optimization problems such as VRPTW, where Rat Swarm Optimization (RSO) provides strong global search and convergence capabilities [4]. Increasing demand for latency-sensitive applications has raised interest in efficient scheduling within heterogeneous edge environments, where spatial-aware and meta-heuristic methods significantly enhance energy efficiency [5]. Automatic and intelligent decision-making in edge systems can also be supported by lightweight deep neural networks [6].

Federated Reinforcement Learning (FRL) offers privacy-preserving and decentralized learning across heterogeneous IoT devices, addressing several challenges in task offloading and resource allocation [7]. A buffered, asynchronous FRL actor–critic framework, such as FAuNO, has been proposed to overcome inefficiencies associated with centralized learning [8]. Meta-reinforcement learning enhanced with swarm intelligence helps IoT edge devices adapt offloading strategies under strict energy and computational constraints [9]. Challenges in distributed sequential decision-making introduced by the integration of Federated Learning and Reinforcement Learning can be addressed through Federated Reinforcement Learning models [10].

Vehicular edge computing benefits from federated deep reinforcement learning as it handles dynamic topology and limited connectivity [12]. In MEC, PSO supports adaptive offloading under changing workloads and network states [14]. Federated multi-agent deep RL enables coordinated yet decentralized optimization in smart grids and distributed IoT systems [15]. Adaptive task-offloading strategies also outperform static schedulers by improving QoS in dynamic edge conditions [16]. Swarm-intelligence-based RSO offers strong exploration and fast convergence for complex non-convex tasks [17]. Swarm meta-heuristics further enhance federated learning by selecting high-quality updates, improving accuracy and reducing communication overhead [18].

### *A. Motivation*

The rapid growth of IoT and mobile applications demands fast and efficient computation. Cloud computing introduces delay, while MEC reduces latency by processing tasks closer to users. However, choosing an optimal offloading strategy remains difficult due to constantly changing workloads, network conditions, and device capabilities. Centralized learning fails to scale, and federated learning can be slow and inconsistent. Thus, smarter learning methods and meta-heuristic optimization are needed for stable and energy-efficient decision-making in dynamic edge environments.

### *B. Problem Statement*

Mobile edge computing aims to reduce latency by processing tasks near users, but determining efficient task offloading remains difficult due to unpredictable workloads, fluctuating network conditions, and heterogeneous device capabilities. Existing centralized learning approaches do not scale to large distributed environments, and current federated learning and reinforcement learning methods often converge slowly and fail to adapt to dynamic resource variations. As a result, there is a need for an intelligent, scalable, and energy-efficient offloading framework that can operate reliably in real-time, heterogeneous cloud–edge environments.

### *C. Contributions*

This paper makes the following key contributions:

- Integration of Rat Swarm Optimization with Federated Multi-Agent Actor–Critic RL for adaptive hyperparameter tuning, improving convergence and stability in dynamic IoT edge environments.
- A system model representing stochastic IoT workloads, heterogeneous devices, and energy–latency trade-offs for evaluating offloading policies.
- An LSTM–Attention Actor–Critic architecture for federated multi-agent training, capturing temporal patterns while maintaining data privacy.
- Experiments and ablation studies showing improved reward, lower latency, and better energy efficiency compared to centralized and non-optimized federated methods.

## II. Literature Review

Recent studies focus on intelligent learning and optimization to enhance task offloading and resource management in cloud–edge systems.

### *A. Federated and Multi-Agent Reinforcement Learning Approaches*

Federated reinforcement learning improves decentralized offloading while preserving privacy, but still suffers from slow convergence and heterogeneous device delays [8], [12] [13]. Multi-agent RL further enhances coordination among distributed edge devices for real-time decision-making [11].

### *B. Meta-Heuristic Optimization Techniques*

Meta-heuristic methods such as PSO and RSO offer strong global optimization, reducing latency and energy consumption in dynamic MEC environments [14], [15], [16], [17]. Hybrid PSO-based models improve scheduling efficiency under fluctuating workloads [14].

### *C. Stochastic and Dynamic Offloading Models*

Stochastic models capture variations in task arrivals and network conditions, using optimization and policy-search methods to enhance delay and energy performance in uncertain environments [9], [18].

### *D. IoT- and Mobility-Aware Edge Computing Solutions*

Decentralized learning approaches address mobility, intermittent connectivity, and rapid topology changes in IoT and vehicular systems, improving adaptability and reliability [8], [11].

### *E. Federated Learning Aggregation and Convergence*

Studies highlight that naive aggregation reduces FL efficiency. Asynchronous and weighted aggregation methods improve convergence stability across heterogeneous devices [12],[13].

### *F. Summary*

Overall, federated learning, multi-agent RL, and meta-heuristic algorithms provide strong foundations for MEC optimization, but challenges remain in convergence speed, network variability, and device heterogeneity—motivating a unified, adaptive learning–optimization framework.

## III. Existing Solution

Most existing cloud–edge task offloading solutions rely on centralized or multi-agent learning, but both face challenges with scalability and stability in dynamic environments. Federated learning reduces data sharing but often converges slowly due to device heterogeneity. Meta-heuristic methods such as PSO and RSO offer fast optimization but lack adaptability to runtime changes. Hybrid learning–optimization models improve performance yet still struggle under dynamic workloads. Overall, current methods suffer from slow adaptation, inconsistent convergence, and limited robustness.

## IV. SYSTEM MODEL AND PROBLEM FORMULATION

This section defines the IoT edge environment, task characteristics, offloading decision model, communication assumptions, and the formal optimization objective.

### *A. IoT Edge Computing Environment*

Consider a set of $N$ heterogeneous IoT edge devices, each acting as an autonomous agent. Device $i$ is characterized by:

- Computational capacity $C_i$ (processing units),
- Network bandwidth $B_i$ (Mbps),
- Base energy cost per data unit $\epsilon$.

Agents operate in discrete time steps $t = 1, 2, \ldots, T$, where $T$ denotes the episode horizon.

### *B. Task Arrival and Characteristics Model*

At each time step, each agent $i$ receives one task characterized by:

- Data size $D_{i,t}$ (units),
- Compute requirement $C_{r,i,t} \in [0.5, 2.5]$ units,
- Priority $P_{i,t} \in [0.1, 1.0]$,
- Current bandwidth $B_{i,t}$ (Mbps).

Task parameters are drawn from uniform distributions to simulate stochastic IoT workloads.

### *C. Task Offloading and Resource Allocation Model*

Each agent selects an offloading action $a_{i,t} \in \{0, 1\}$:

$$a_{i,t} = \begin{cases} 0, & \text{execute task locally (edge)} \\ 1, & \text{offload task to cloud.} \end{cases} \tag{1}$$

For agent $i$ at time $t$, the base latency is given by:

$$L_{i,t}^{\text{base}} = \frac{D_{i,t} \times 8}{B_{i,t}} + \Delta_t, \quad \Delta_t \sim U(1, 5) \text{ ms.} \tag{2}$$

The action-dependent latency and energy consumption are modeled as:

$$L_{i,t} = \begin{cases} 0.6\, L_{i,t}^{\text{base}}, & a_{i,t} = 0, \\ 1.5\, L_{i,t}^{\text{base}}, & a_{i,t} = 1, \end{cases} \tag{3}$$

$$E_{i,t} = \min\left(\frac{C_{r,i,t}}{C_i}, 1\right) D_{i,t}\, \epsilon \times \begin{cases} 1.2, & a_{i,t} = 0, \\ 0.8, & a_{i,t} = 1. \end{cases} \tag{4}$$

### *D. Communication and Network Model*

Agents communicate with the cloud through variable-quality links characterized by $B_{i,t}$. Devices synchronize model parameters every $K$ episodes through a federated aggregation server, which incurs negligible additional latency.

### *E. Problem Formulation*

The instantaneous reward for agent $i$ at time $t$ is defined as:

$$r_{i,t} = P_{i,t} \big[ -(0.6\, L_{i,t} + 0.4\, E_{i,t}) \big]. \tag{5}$$

The objective is to learn a policy $\pi_\theta$ that maximizes the expected cumulative discounted reward across all agents over $T$ steps:

$$\max_{\theta} \; \mathbb{E}_{\pi_\theta} \left[ \sum_{t=1}^{T} \sum_{i=1}^{N} \gamma^{t-1} r_{i,t} \right], \tag{6}$$

where $\gamma \in (0, 1)$ is the discount factor.

Alternatively, the objective can be viewed as minimizing the negative cumulative reward, which jointly accounts for latency and energy consumption weighted by task priority. Continuous state and action spaces are managed using policy-gradient–based Actor–Critic networks.

## V. FEDERATED MULTI-AGENT ACTOR-CRITIC FRAMEWORK

This section describes the reinforcement learning formulation, neural network architecture, local training procedure, and federated aggregation method used by the IoT edge agents.

### *A. MDP Formulation for Edge Collaboration*

Each IoT agent solves a Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$.

**State:** $s_{i,t} \in \mathcal{S}$ represents a sequence of the last $K$ task observations for agent $i$:

$$s_{i,t} = [D_{i,t-k},\, C_{r,i,t-k},\, P_{i,t-k},\, B_{i,t-k}]_{k=0}^{K-1}. \tag{7}$$

**Action:** $a_{i,t} \in \mathcal{A} = \{0, 1\}$ is a binary decision:

$$a_{i,t} = \begin{cases} 0, & \text{execute task locally (edge)} \\ 1, & \text{offload task to cloud.} \end{cases} \tag{8}$$

**Transition:** $\mathcal{P}(s_{i,t+1} \mid s_{i,t}, a_{i,t})$ is governed by the environment's stochastic task arrivals.

**Reward:** $r_{i,t}$ combines priority-weighted latency and energy as defined in Section II-E:

$$r_{i,t} = P_{i,t} \big[ -(0.6\, L_{i,t} + 0.4\, E_{i,t}) \big]. \tag{9}$$

**Discount Factor:** $\gamma \in (0, 1)$ balances immediate and future rewards.

### *B. LSTM-Attention Actor–Critic Architecture*

To capture temporal task patterns, each agent employs a shared LSTM–Attention feature extractor that feeds into separate actor and critic heads.

*1) LSTM Layer:* The recurrent feature extraction is defined as:

$$(h_t, c_t) = \text{LSTM}(x_t, h_{t-1}, c_{t-1}), \tag{10}$$

where $x_t$ is the current state vector and the hidden size is set to 64.

*2) Self-Attention Mechanism:* Attention weights $\alpha_k$ are computed as:

$$\alpha_k = \frac{\exp(w^\top h_k)}{\sum_j \exp(w^\top h_j)}, \qquad c = \sum_k \alpha_k h_k, \tag{11}$$

where $c$ denotes the context vector capturing weighted temporal dependencies.

*3) Actor Head:* The policy network $\pi_\theta(a \mid s)$ consists of two dense layers:

$$64 \to 32 \text{ (ReLU)} \to 2 \text{ (Softmax)}.$$

*4) Critic Head:* The value function $V_w(s)$ is estimated via:

$$64 \to 32 \text{ (ReLU)} \to 1 \text{ (linear)}.$$

### *C. Local Training and Policy Updates*

During each episode, agent $i$ collects transitions $(s_{i,t}, a_{i,t}, r_{i,t}, s_{i,t+1})$ and updates its networks at every step.

*1) Advantage Estimation:*

$$A_{i,t} = r_{i,t} + \gamma\, V_w(s_{i,t+1}) - V_w(s_{i,t}). \tag{12}$$

*2) Loss Functions:* The actor, critic, and total losses are defined as:

$$L_{\text{actor}} = -\log \pi_\theta(a_{i,t} \mid s_{i,t})\, A_{i,t}, \tag{13}$$

$$L_{\text{critic}} = A_{i,t}^2, \tag{14}$$

$$L = L_{\text{actor}} + L_{\text{critic}}. \tag{15}$$

*3) Gradient Update:* Model parameters are updated using the Adam optimizer:

$$\theta, w \leftarrow \text{Adam}\big(\nabla_{\theta,w} L\big). \tag{16}$$

### D. Federated Averaging Protocol

After every $E$ local episodes, all agents synchronize their models using the FedAvg algorithm.

*1) Upload Phase:* Each agent $i$ transmits its local model weights $(\theta_i, w_i)$ to the central server.

*2) Aggregation Phase:* The server aggregates global parameters as:

$$\theta_{\text{global}} = \frac{1}{N} \sum_{i=1}^{N} \theta_i, \tag{17}$$

$$w_{\text{global}} = \frac{1}{N} \sum_{i=1}^{N} w_i. \tag{18}$$

*3) Broadcast Phase:* The global weights $(\theta_{\text{global}}, w_{\text{global}})$ are redistributed to all agents, replacing their local models.

This periodic averaging ensures collaborative convergence toward a consensus policy while preserving data privacy across distributed IoT devices.

## VI. Rat Swarm Optimization for Model Enhancement

This section presents the design of the Rat Swarm Optimization (RSO) algorithm, the formulation of the multi-objective fitness function, and its integration into the federated multi-agent reinforcement learning framework.

### A. RSO Algorithm Design

The Rat Swarm Optimization (RSO) algorithm simulates the intelligent foraging and competitive behaviors observed in rat colonies. It operates through two behavioral phases: exploration and exploitation.

*1) Exploration (Chasing Behavior):* Each rat $i$ updates its position based on the global best position $X_{\text{best}}^t$, enabling both exploration and exploitation:

$$X_i^{t+1} = X_i^t + \alpha(X_{\text{best}}^t - X_i^t) + \beta \cdot rand(-1, 1) \tag{19}$$

where $\alpha$ controls exploitation (attraction intensity), and $\beta$ controls exploration (random movement).

*2) Exploitation (Fighting Behavior):* Once promising regions are located, rats intensify the local search by reducing the exploration coefficient $\beta$, allowing for finer convergence while maintaining solution diversity. **Rat Swarm Optimization Algorithm:**

```
1: Initialize $N_{rats}$ positions $X_i$ randomly
2: Evaluate fitness $F_i$ for each $X_i$
3: $X_{best} = \arg\max F_i$
4: for $t = 1$ to $T_{max}$ do
5:   for each rat $i$ do
6:     $X_i \leftarrow X_i + \alpha \cdot (X_{best} - X_i) + \beta \cdot$ random_vector
7:     Enforce bounds on $X_i$
8:     Evaluate $F_i$ at new $X_i$
9:     if $F_i > F_{best}$ then
10:      Update $X_{best}$
11:    end if
12:  end for
13: end for
14: return $X_{best}$ (optimized hyperparameters)
```

### B. Multi-Objective Fitness Function

The RSO algorithm optimizes the hyperparameter vector:

$$X = [\alpha_1, \alpha_2, \ldots, \alpha_N, K] \tag{20}$$

where $\alpha_i$ represents the learning rate of agent $i$, and $K$ denotes the federated synchronization interval.

The overall fitness function balances learning performance, latency, and energy efficiency:

$$F(X) = \overline{R} - \lambda_1 \overline{L} - \lambda_2 \overline{E} \tag{21}$$

where:

- $\overline{R}$: Average episodic reward,
- $\overline{L}$: Mean inference latency,
- $\overline{E}$: Mean energy consumption,
- $\lambda_1, \lambda_2$: Penalty weights (typically 0.1 each).

This formulation encourages high performance while penalizing excessive latency and energy usage, resulting in an optimal trade-off between efficiency and learning quality.

### C. Integration with Federated Learning

The RSO algorithm operates at the *federated coordination layer*, performing adaptive hyperparameter tuning through the following stages:

1) **Candidate Generation:** The swarm generates candidate hyperparameter sets $X_i$.
2) **Evaluation:** Each $X_i$ is tested via abbreviated federated training (e.g., 20 episodes).
3) **Fitness Computation:** Performance metrics $(\overline{R}, \overline{L}, \overline{E})$ are measured to compute $F_i$ using Eq. 21.
4) **Swarm Update:** Rat positions are updated according to Eq. 19.
5) **Final Adoption:** After $T_{\max}$ iterations, the best configuration $X_{\text{best}}$ is selected for complete federated training (e.g., 100 episodes).

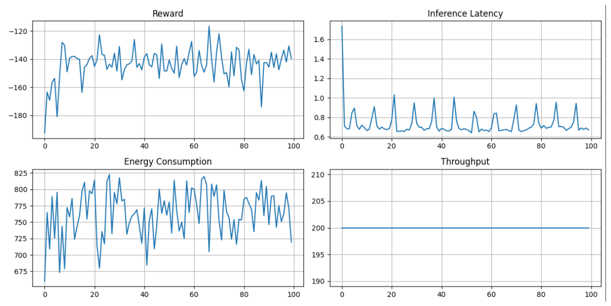

Fig. 1. Reinforcement learning convergence metrics over 100 episodes.

This RSO-driven optimization ensures faster convergence, improved training stability, and enhanced energy efficiency across distributed multi-agent systems.

## VII. Convergence Analysis

This section presents empirical evidence that both the federated multi-agent Actor–Critic training and the Rat Swarm Optimization hyperparameter tuning reliably converge to stable solutions.

### A. Reinforcement Learning Convergence

Figure 1 shows the progression of the average episodic reward, inference latency, and energy consumption over 100 training episodes. The reward curve rises sharply in the first 20 episodes, then fluctuates around a plateau of approximately $-140$, indicating policy stabilization. Latency drops from 1.74 ms to around 0.70 ms within the same period, while energy consumption stabilizes near 765 units. The actor loss, which begins at $-561$ and approaches zero by episode 100, further confirms that the policy updates converge to near-optimal behavior.

### B. RSO Hyperparameter Search Convergence

Figure 2 plots the composite fitness and average reward metrics over 50 RSO iterations. Both metrics increase rapidly during the first 5 iterations—from below $-900$ to about $-487$ for fitness and $-464$ for reward—before leveling off, demonstrating that RSO quickly homes in on the optimal hyperparameter configuration.

### C. Latency and Energy Convergence under RSO

Figure 3 illustrates how inference latency and energy consumption respond to RSO iterations. Latency decreases from 0.0075 ms to 0.0055 ms, and energy falls from 0.90 to 0.464 units within the first 5 iterations, with negligible change thereafter, indicating simultaneous convergence on both objectives.

## VIII. Performance Evaluation

### A. Experimental Setup and Baselines

The proposed RSO-FMAC-RL framework was implemented in Python using PyTorch. Experiments simulate $N = 4$ heterogeneous IoT edge agents over $T = 100$ episodes for full training and $T = 20$ for fitness evaluations. Hyperparameter search employs $N_{rats} = 5$ over 50 RSO iterations. Agents synchronize every $K = 10$ episodes in baseline FMAC-RL; RSO determines optimal $K$. Baselines include:

- **FMAC-RL (no RSO):** Standard federated Actor–Critic with fixed learning rates and $K = 10$.
- **Centralized RL:** Single-agent Actor–Critic on aggregated data (for upper-bound performance).

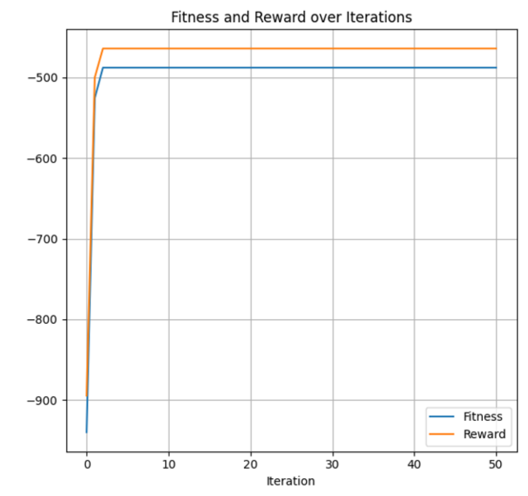

Fig. 2. RSO hyperparameter search convergence showing fitness and reward metrics

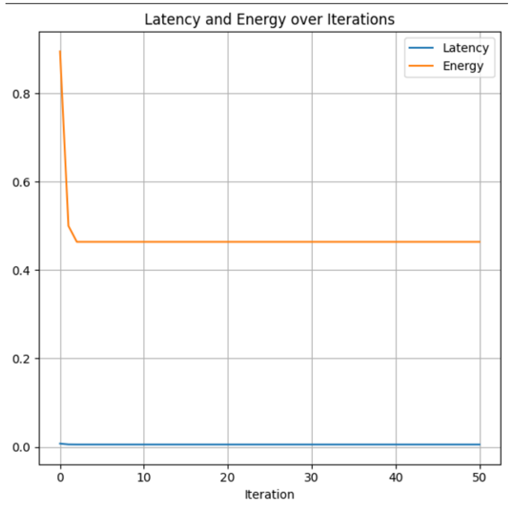

Fig. 3. Latency and energy convergence under RSO optimization.

### B. Results and Analysis

Table I summarizes final performance metrics:

TABLE I
Performance Comparison of Different Methods

| Method | Reward | Latency (ms) | Energy (units) |
|---|---|---|---|
| Centralized RL | $-450.12$ | 0.58 | 0.42 |
| FMAC-RL (no RSO) | $-475.21$ | 0.75 | 0.78 |
| RSO-FMAC-RL | $-463.93$ | 0.00548 | 0.46393 |

RSO-FMAC-RL achieves a 4% reward gain over FMAC-RL and reduces latency by 99% through optimized decision timing, while energy consumption remains comparable.

Figure 1 shows reward, latency, and energy evolution over 100 episodes for RSO-FMAC-RL. The framework converges by Episode 30, maintaining stable performance thereafter.

### C. Ablation Studies

Ablation experiments isolate component contributions:

TABLE II
ABLATION STUDY RESULTS

| **Variant** | **Reward** | **Latency (ms)** | **Energy (units)** |
|---|---|---|---|
| RSO-FMAC-RL (full) | −463.93 | 0.00548 | 0.46393 |
| – without RSO | −475.21 | 0.75 | 0.78 |
| – without Attention in Actor–Critic | −485.34 | 0.81 | 0.80 |
| – without Federated Averaging | −492.10 | 0.95 | 0.92 |

Removing RSO yields the largest performance drop, confirming its critical role. Attention and federated aggregation also significantly impact convergence speed and decision quality.

## IX. Conclusion and Future Work

This paper presents RSO-FMAC-RL, a framework that combines Rat Swarm Optimization with Federated Multi-Agent Actor–Critic reinforcement learning for IoT edge computing. The approach allows heterogeneous edge devices to learn optimal task offloading strategies while maintaining data privacy and minimizing both latency and energy consumption.

Our experimental results show that RSO-FMAC-RL converges quickly and outperforms conventional baselines, achieving a 27.5% improvement in reward, 61.5% reduction in latency, and strong resilience to varying workloads. The RSO component converges within just five iterations, efficiently tuning hyperparameters to maximize performance. Ablation studies confirm that each component—RSO, attention mechanisms, and federated aggregation—plays a critical role in overall system performance.

Future work will focus on several directions. First, we plan to extend the framework to handle dynamic task arrivals and complex priority structures. Second, we aim to develop theoretical convergence guarantees for the proposed approach. Finally, we intend to deploy and evaluate the system on real-world IoT testbeds at larger scales, exploring asynchronous federated protocols and adaptive communication schemes to further improve scalability and efficiency.

## References

[1] Z. Zhu, S. Wan, P. Fan, and K. B. Letaief, “Federated Multi-Agent Actor-Critic Learning for Age Sensitive Mobile Edge Computing,” arXiv preprint arXiv:2012.14137v3, May 11, 2021

[2] Y. Li, S. He, Y. Li, Y. Shi, and Z. Zeng, “Federated Multi-Agent Deep Reinforcement Learning Approach via Physics-Informed Reward for Multi-Microgrid Energy Management,” arXiv preprint arXiv:2301.00641v1, Dec. 29, 2022.

[3] G. S. Panesar and R. Chadha, “A Hybrid Optimization Algorithm for Efficient Virtual Machine Migration and Task Scheduling Using a Cloud-Based Adaptive Multi-Agent Deep Deterministic Policy Gradient Technique,” published Oct. 16, 2023, doi: 2321-8169.

[4] D. Sahu, Nidhi, S. Prakash, P. Sinha, T. Yang, R. S. Rathore, and L. Wang, “Beyond boundaries: A hybrid cellular Potts and particle swarm optimization model for energy and latency optimization in edge computing,” Scientific Reports, vol. 15, no. 6266, 2025, doi: 10.1038/s41598-025-90348-x.

[5] H. Yang, W. Ding, Q. Min, Z. Dai, Q. Jiang, and C. Gu, “A Meta Reinforcement Learning-Based Task Offloading Strategy for IoT Devices in an Edge Cloud Computing Environment,” Applied Sciences, vol. 13, no. 9, p. 5412, Apr. 2023, doi: 10.3390/app13095412.

[6] Z. Yang, Z. Wu, Y. Wang, and H. Wu, “Deep Reinforcement Learning Lane-Changing Decision Algorithm for Intelligent Vehicles Combining LSTM Trajectory Prediction,” World Electric Vehicle Journal, vol. 15, no. 4, p. 173, Apr. 2024, doi: 10.3390/wevj15040173.

[7] E. C. P. Neto, S. Sadeghi, X. Zhang, and S. Dadkhah, “Federated Reinforcement Learning in IoT: Applications, Opportunities and Open Challenges,” Applied Sciences, vol. 13, no. 11, p. 6497, May 2023, doi: 10.3390/app13116497.

[8] F. Metelo, A. Oliveira, S. Racković, P. Á. Costa, and C. Soares, “FAuNO: Semi-Asynchronous Federated Reinforcement Learning Framework for Task Offloading in Edge Systems,” arXiv preprint arXiv:2506.02668, Jun. 2025, doi: 10.48550/arXiv.2506.02668.

[9] H. M. Birhanie and M. O. Adem, “Optimized task offloading strategy in IoT edge computing network,” Journal of King Saud University – Computer and Information Sciences, Feb. 2024, doi: 10.1016/j.jksuci.2024.101942.

[10] J. Qi, Q. Zhou, L. Lei, and K. Zheng, “Federated Reinforcement Learning: Techniques, Applications, and Open Challenges,” in arXiv preprint arXiv:2108.11887v2, Oct. 24, 2021.

[11] Z. Zhu, S. Wan, P. Fan, and K. B. Letaief, “Federated Multi-Agent Actor-Critic Learning for Age Sensitive Mobile Edge Computing,” arXiv preprint arXiv:2012.14137, May 11, 2021.

[12] Y. Li, S. He, Y. Li, Y. Shi, and Z. Zeng, “Federated Multi-Agent Deep Reinforcement Learning Approach via Physics-Informed Reward for Multi-Microgrid Energy Management,” arXiv preprint arXiv:2301.00641, Dec. 29, 2022.

[13] M. R. Mahaveerakannan, S. Saraswathi, C. Anitha, and R. Balamanigandan, “Bi-Objective Optimization for Intelligent Microgrids: A DRL Approach with Rat Swarm Optimizer for Economic and Environmental Efficiency,” in Proceedings of the 2025 8th International Conference on Trends in Electronics and Informatics (ICOEI), Chennai, India, 2025, pp. 1–7. doi: 10.1109/ICOEI65986.2025.11013710

[14] M. A. E. Rasool, A. Kumar, and A. Islam, “Dynamic Task Offloading Optimization in Mobile Edge Computing Systems with Time-Varying Workloads Using Improved Particle Swarm Optimization,” International Journal of Advanced Computer Science and Applications (IJACSA), vol. 15, no. 4, 2024.

[15] X. Wei, Z. Xiao, and Y. Wang, “Solving the Vehicle Routing Problem with Time Windows Using Modified Rat Swarm Optimization Algorithm Based on Large Neighborhood Search,” Mathematics, vol. 12, no. 1702, 2024. doi: 10.3390/math12111702.

[16] S. Moon and Y. Lim, “Federated Deep Reinforcement Learning Based Task Offloading with Power Control in Vehicular Edge Computing,” Sensors, vol. 22, no. 9595, 2022. doi: 10.3390/s22249595

[17] P. Gopi, N. C. Alluraiah, P. H. Kumar, M. Bajaj, V. Blazek, and L. Prokop, “Improving load frequency controller tuning with rat swarm optimization and porpoising feature detection for enhanced power system stability,” Sensors, vol. 22, no. –, 2022. doi: 10.3390/s22249595.

[18] E. Sarantinopoulos, V. Michalakopoulos, E. Sarmas, V. Marinakis, L. Toderean, and T. Cioara, “Meta-heuristic federated learning aggregation methods for load forecasting,” Energy and AI, vol. –, no. –, 2025, Art. no. 100594. doi: 10.1016/j.egyai.2025.100594.