DAGMARL-Fog: Graph-Aware Multi-Agent Learning for Adaptive and Sustainable Fog Task Scheduling

Gurpreet Singh Chhabra[1], *Subhendu Kumar Pani*[2]

[1]Postdoctoral Researcher, Lincoln University College, 47301, Petaling Jaya, Selangor Darul Ehsan, Malaysia;

[2]Department of Computer Science and Engineering, Krupajal Engineering College, Biju Patnaik University of Technology (BPUT), Odisha, India.

Email IDs: pdf.gurpreet@lincoln.edu.my, skpani.utkal@gmail.com.

Abstract: Latency-dependent IoT systems can be supported by fog computing; however, problems such as high control overhead, late synchronization, and non-scalability arise from centralized scheduling approaches. This paper introduces DAGMARL-Fog, a decentralized framework of multi-agent reinforcement learning with graph-aware state representation for scheduling tasks within IoT, fog, and cloud environments. The proposed framework uses graph representation of states, feasibility-filtered MARL, consensus-based neighborhood choice, and objective weight adjustment to improve latency, energy, load balance, reliability, and carbon emission. Simulations based on iFogSim2, latency modeling through EdgeCloudSim, and Google Cluster Trace achieve 132 ms latency, 7900 J energy, 95.6% deadline fulfillment rate, and 0.94 fairness index.

Keywords: Fog computing; task scheduling; multi-agent reinforcement learning; graph neural networks; IoT; latency optimization; energy efficiency; reliability; carbon-aware computing.

Introduction and Related Work

The heterogeneous and highly bursty tasks generated from IoT applications need to be efficiently processed within edge, fog, and cloud layers. Task scheduling becomes an inefficient solution under such circumstances since it requires significant control overhead, creates single-point dependency, causes delay in state synchronization, and has little scalability. While fog computing lessens dependence on cloud computing, the scheduler needs to make decisions on where each task needs to be executed locally or forwarded to neighboring fog nodes or to the cloud.

There are existing solutions dealing with certain aspects of the problem. HEFT is a classical heuristic approach for heterogeneous task scheduling but not specifically for fog decentralization [1]. On the other hand, NSGA-II deals with multi-objective optimization but can become too costly when it comes to dynamic workloads [2]. DQN and PPO are examples of deep reinforcement learning approaches providing adaptive scheduling but do not fully represent the cooperation of distributed fog nodes [3], [4]. MADDPG offers a multi-agent decision-making but can potentially lead to high communication and coordination overhead [5]. Graph learning algorithms allow modeling relationships between nodes effectively but still need to be integrated with feasibility constraints and light-weight coordination [6]. Simulation environments like iFogSim2 and

EdgeCloudSim can be used for the evaluation of resource allocation and latency behavior in fog and edge systems [7], [8].

In order to combine all the above-mentioned advantages, the presented paper proposes DAGMARL-Fog, which is a decentralized graph-aware MARL framework for task scheduling within IoT-fog-cloud system. In this framework, fog nodes act as agents, while fog topology is modeled with the help of a graph. It allows filtering out infeasible actions for the task and updating the weight values of objectives depending on workload situation. The purpose of DAGMARL-Fog is joint minimization of latency, energy consumption, load balance, reliability loss, and carbon emissions.

System Architecture and Mathematical Formulation

The suggested architecture for the fog scheduling mechanism has been illustrated in Fig. 1 below. The system comprises IoT devices, cooperative fog nodes, and a distant cloud. The IoT devices generate tasks which have varied resource requirements as well as deadlines. The fog node functions as a scheduling agent in the system and makes the scheduling decision based on its local information and that of its neighbors.

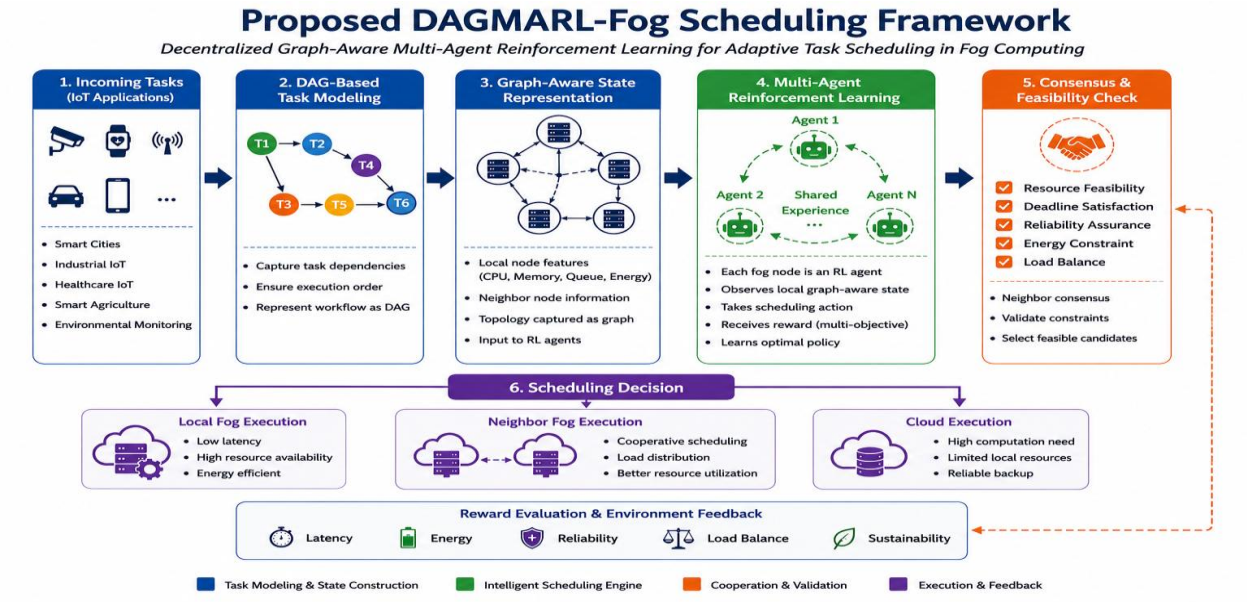


Fig. 1. System architecture of the proposed DAGMARL-Fog framework

Let $\mathcal{T} = \{\tau_1, \tau_2, \dots, \tau_N\}$ denote the set of IoT tasks and $\mathcal{F} = \{f_1, f_2, \dots, f_M\}$ denote the set of fog nodes. The cloud server is denoted by c. The task assignment variable is defined as:

$$x_{ij} = \begin{cases} 1, & \text{if task } \tau_i \text{ is assigned to fog node } f_j, \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

The cloud offloading variable is defined as

$$y_i = \begin{cases} 1, & \text{if task } \tau_i \text{ is offloaded to the cloud,} \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

The multi-objective scheduling cost is formulated as

$$\min Z = \alpha C^L + \beta C^E + \gamma C^B + \delta C^R + \epsilon C^C \quad (3)$$

subject to

$$\alpha + \beta + \gamma + \delta + \epsilon = 1, \alpha, \beta, \gamma, \delta, \epsilon \geq 0 \quad (4)$$

Here, C^L is the average normalized latency, C^E is the normalized computation and communication energy, C^B is the normalized load-imbalance cost, C^R is the reliability loss, and C^C is the normalized carbon-emission cost. The objective guides the scheduler to reduce delay and energy while avoiding overloaded, unreliable, and carbon-intensive execution nodes.

The formulation is constrained by assignment, resource, communication, deadline, reliability, queue-stability, carbon, and neighbor-forwarding conditions. Each task must be processed exactly once as:

$$\sum_{j=1}^{M} x_{ij} + y_i = 1, \forall \tau_i \in \mathcal{T} \quad (5)$$

The binary decision constraint is defined as:

$$x_{ij}, y_i \in \{0,1\} \quad (6)$$

The fog-node CPU capacity constraint is:

$$\sum_{i=1}^{N} x_{ij}\, r_i^{cpu} \leq R_j^{cpu}, \forall f_j \in \mathcal{F} \quad (7)$$

Similarly, memory and storage constraints are expressed as:

$$\sum_{i=1}^{N} x_{ij}\, r_i^{mem} \leq R_j^{mem}, \sum_{i=1}^{N} x_{ij}\, r_i^{sto} \leq R_j^{sto} \quad (8)$$

The bandwidth constraint is defined as:

$$\sum_{i=1}^{N} x_{ij}\, b_i \leq B_j \quad (9)$$

The deadline condition is:

$$L_i \leq D_i, \forall \tau_i \in \mathcal{T} \quad (10)$$

where L_i is the final task latency and D_i is the task deadline. Queue stability at each fog node is maintained as:

$$\lambda_j < \mu_j \quad (11)$$

where λ_j and μ_j are the task arrival rate and service rate of fog node f_j, respectively. Reliability-aware scheduling is enforced through $R_j \geq R_i^{min}$, where R_j is the reliability of fog node f_j, and R_i^{min} is the minimum reliability required by task τ_i. Neighbor forwarding is permitted only when a cooperative link exists between two fog nodes. These constraints convert task scheduling from a simple placement problem into a constrained reliability-aware and sustainability-aware optimization problem.

Proposed Methodology

The DAGMARL-Fog is made up of four highly interlinked components: state construction using graphs, multi-agent reinforcement learning, consensus scheduling, and adaptive weight update. Figure 1 provides a graphic overview of the process of decentralized graph-aware scheduling.

Each of the fog nodes detects the state for itself and receives compact descriptors from neighboring fog nodes. These include CPU usage, length of queue, memory usage, bandwidth, reliability, energy consumption, and carbon intensity.

The multi-agent reinforcement learning (MARL) agent learns to determine whether the incoming task must be run on the local fog node, transferred to one of the neighboring fog nodes, or uploaded to the cloud. Prior to execution, a feasibility filter filters out the actions that are not possible due to CPU usage limits, memory, storage, bandwidth, deadline, queue stability, reliability, and neighbor link limitations.

The reward function is aligned with the multi-objective formulation and is expressed as

$$R_t = -(w_L C_t^L + w_E C_t^E + w_B C_t^B + w_R C_t^R + w_C C_t^C) - \Omega_t \quad (12)$$

where w_L, w_E, w_B, w_R, w_C are adaptive objective weights, and Ω_t is the constraint violation penalty. The reward penalizes high latency, energy consumption, load imbalance, reliability loss, carbon emission, and infeasible actions.

Consensus Scheduling applies to cases where task forwarding needs to be done to neighbors. Ranking of neighboring fog nodes is done based on the utilization, queue length, bandwidth, and reliability descriptors in order to avoid forwarding to overloaded and unreliable neighbors. Adaptive weight update assigns a higher weight to a degraded objective when there is a change in workload. In particular, the latency weight is increased in case of bursty task arrivals, reliability weight is increased during failures, and carbon weight is increased during sustainability-sensitive periods.

Experimental Setup

This evaluation is done with the use of iFogSim2 and EdgeCloudSim-based latency modeling [7], [8]. This simulation involves the architecture of IoT-fog-cloud model. There are 500 to 10,000 IoT devices and 10 to 500 fog nodes. The topology of fog layer is sparse cooperative graph. Each fog node has 2 to 8 neighbors. The simulation time is 3,600 seconds, with the first 300 seconds being the warm-up period. There are thirty independent runs.

Task arrival characteristics are taken from Google Cluster Trace, including task arrival, CPU demand and memory demand attributes [9]. There are 1,000 to 100,000 tasks per run. Applications include sensing, healthcare, surveillance, industrial control, and vehicular sensing. There are three levels of workload, namely, low, medium, and high load. Table 1 and 2 give details of simulation parameters, workload, baselines and algorithmic setup.

Table 1. Simulation and infrastructure configuration

Parameter	Setting
Simulator	iFogSim2 with EdgeCloudSim-based latency modeling
Architecture	IoT–Fog–Cloud
IoT devices	500–10,000
Fog nodes	10–500
Fog topology	Sparse cooperative graph
Fog-neighbor degree	2–8 neighboring nodes
Simulation duration	3,600 s, first 300 s excluded

Independent runs	30
Latency range	IoT–fog: 2–15 ms, fog–fog: 2–20 ms, fog–cloud: 50–150 ms

Table 2. Workload, baselines, and algorithmic settings

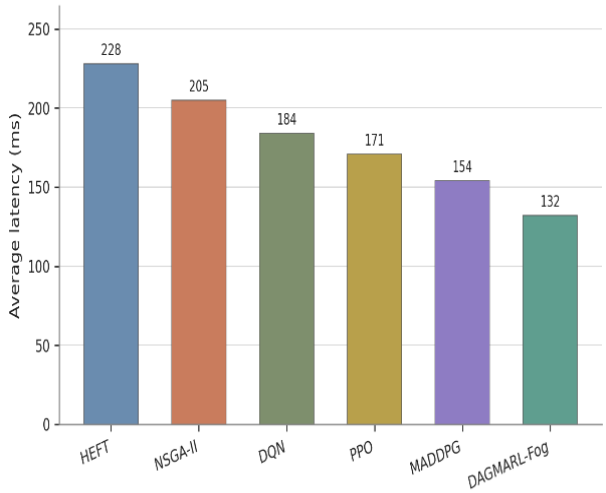
Category	Setting
Primary dataset	Google Cluster Trace
Task volume	1,000–100,000 tasks
Workload levels	Low, medium, high
Compared methods	HEFT, NSGA-II, DQN, PPO, MADDPG, DAGMARL-Fog
Learning rate	0.0005
Discount factor	0.95
Replay buffer	50,000 transitions
Batch size	64
Graph embedding dimension	64
Ablation variants	Without GNN, MARL, consensus, adaptive weights

HEFT serves as the heuristic baseline, NSGA-II is the evolutionary multi-objective optimization technique, DQN and PPO are the single agent Deep Reinforcement Learning techniques, and MADDPG serves as the Multi-Agent Reinforcement Learning comparative model [1]-[5]. Latency, energy consumption, and deadlines served as the primary performance measures, and Jain's fairness index, success rate, communication overhead, and carbon emissions as the secondary measures [10].

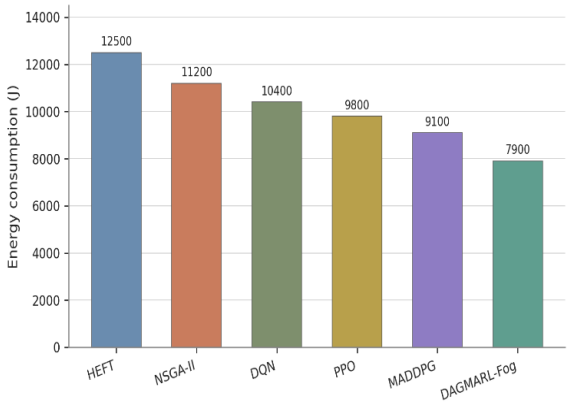
Results and Analysis

The latency performance under heavy workload conditions is shown in Fig. 2a. DAGMARL-Fog obtains the lowest latency of 132 ms. In terms of latency saving, DAGMARL-Fog saves 14.29% latency over MADDPG and 42.10% over HEFT. The results confirm the positive effects of graph-based neighbor modeling and feasibility-filtering in decentralized task placement, which helps reduce waiting time and prevents overload of execution points. In Fig. 2b, the total energy consumption of DAGMARL-Fog is 7900 J, which is the lowest one comparing with the baseline methods. DAGMARL-Fog saves energy by 13.19% over MADDPG and 36.80% over HEFT. It is mainly because the number of unnecessary offloading to the cloud decreases and distributes tasks to less loaded fog nodes. The deadline satisfaction ratio in Fig. 2c shows that DAGMARL-Fog meets 95.6% deadline requirements, which is 3.2% higher than MADDPG and 13.5% higher than HEFT. It proves that the feasibility filtering and adaptive weights are useful in the time-constrained scheduling. The Jain fairness index in Fig. 2d demonstrates the load balancing capability of DAGMARL-Fog. DAGMARL-Fog achieves the highest fairness score of 0.94, improving the fairness index by 5.62% over MADDPG and 20.51% over HEFT. The consensus scheduling makes sure that the nodes will not be allocated repeatedly to heavily loaded fog nodes and make a better utilization distribution. In Fig. 2e, the success rate with 40% fog node failures shows that DAGMARL-Fog has a success rate of 85.0%, which is 8.97% higher than MADDPG and 41.67%

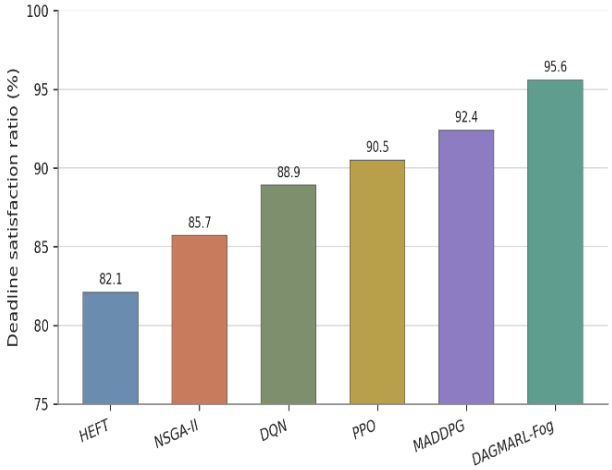
higher than HEFT. The success rate is increased because the reliability-aware rewards and feasibility filtering prevent assignment of tasks to unreliable fog nodes. In the experiment with increasing number of tasks shown in Fig. 2f, when there are 100,000 tasks, DAGMARL-Fog achieves the lowest latency of 186 ms, which is 15.45% lower than MADDPG and 40.00% lower than HEFT. This means that DAGMARL-Fog reduces the global synchronization costs through compact neighbor descriptors while keeping cooperative scheduling. The communication overheads in Fig. 2g show that DAGMARL-Fog has the overhead of 265 KB per each interval for control plane information exchange. It is 36.90% lower than MADDPG and 57.26% lower than NSGA-II, while the overhead is higher than DQN and PPO since DAGMARL-Fog uses neighbor descriptors for cooperation. In the carbon emission experiment in Fig. 2h, DAGMARL-Fog emits only 750 gCO_2e, which is the lowest amount among all compared methods. It reduces the carbon emission by 13.79% over MADDPG and 36.44% over HEFT. This means that energy-aware and carbon-aware scheduling can help minimize the environmental cost in fog computing. The ablation experiment results shown in Fig. 2i indicate that the full DAGMARL-Fog model obtains the lowest latency of 132 ms and outperforms all ablated versions. Removing MARL brings the most degradation, which means that the distributed learning plays an important role in the proposed framework. GNN, consensus scheduling and adaptive weights are also important because they help improve neighborhood awareness, coordination and adaptive objective control. The experiment with weights in Fig. 2j shows that the latency-prioritized W1 setting brings the lowest latency of 124 ms. The energy-prioritized W2 setting reduces energy consumption but increases latency.



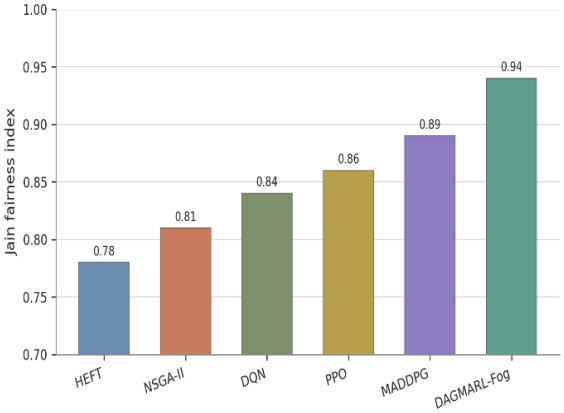
(a)



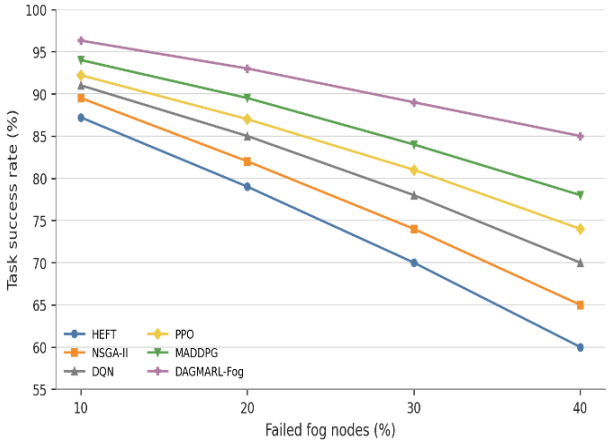
(b)


Deadline satisfaction ratio (%)
100
95
90
85
80
75
82.1
85.7
88.9
90.5
92.4
95.6
HEFT
NSGA-II
DQN
PPO
MADDPG
DAGMARL-Fog

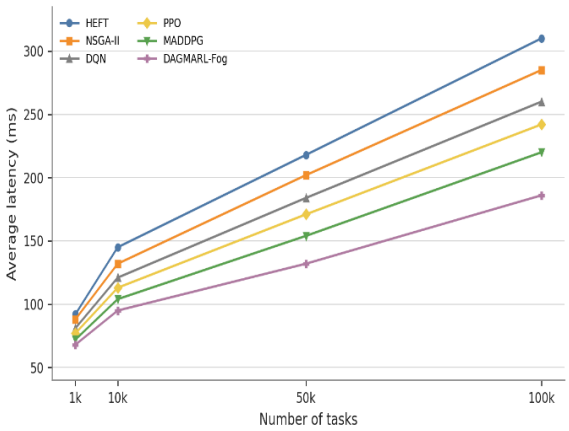
(c)


Jain fairness index
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.78
0.81
0.84
0.86
0.89
0.94
HEFT
NSGA-II
DQN
PPO
MADDPG
DAGMARL-Fog

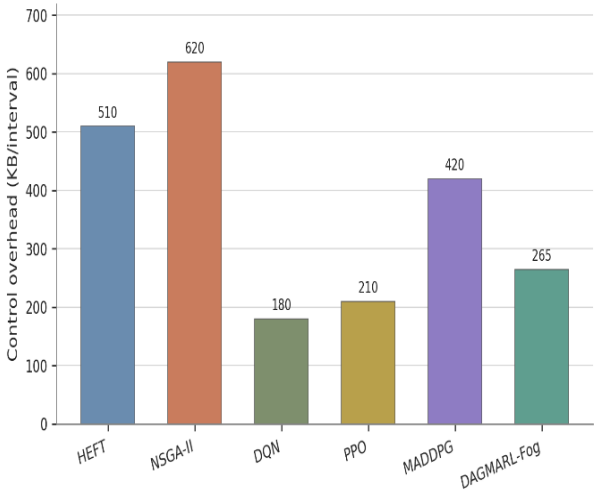
(d)


Task success rate (%)
100
95
90
85
80
75
70
65
60
55
10
20
30
40
Failed fog nodes (%)
HEFT
NSGA-II
DQN
PPO
MADDPG
DAGMARL-Fog

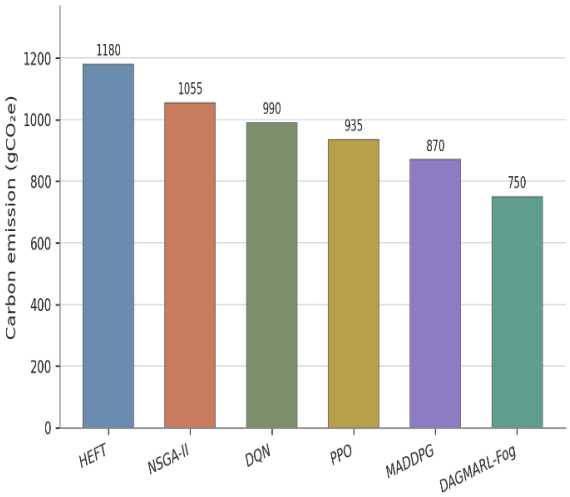
(e)


Average latency (ms)
300
250
200
150
100
50
1k
10k
50k
100k
Number of tasks
HEFT
NSGA-II
DQN
PPO
MADDPG
DAGMARL-Fog

(f)


Control overhead (KB/interval)
700
600
500
400
300
200
100
0
510
620
180
210
420
265
HEFT
NSGA-II
DQN
PPO
MADDPG
DAGMARL-Fog

(g)


Carbon emission (gCO₂e)
1200
1000
800
600
400
200
0
1180
1055
990
935
870
750
HEFT
NSGA-II
DQN
PPO
MADDPG
DAGMARL-Fog

(h)

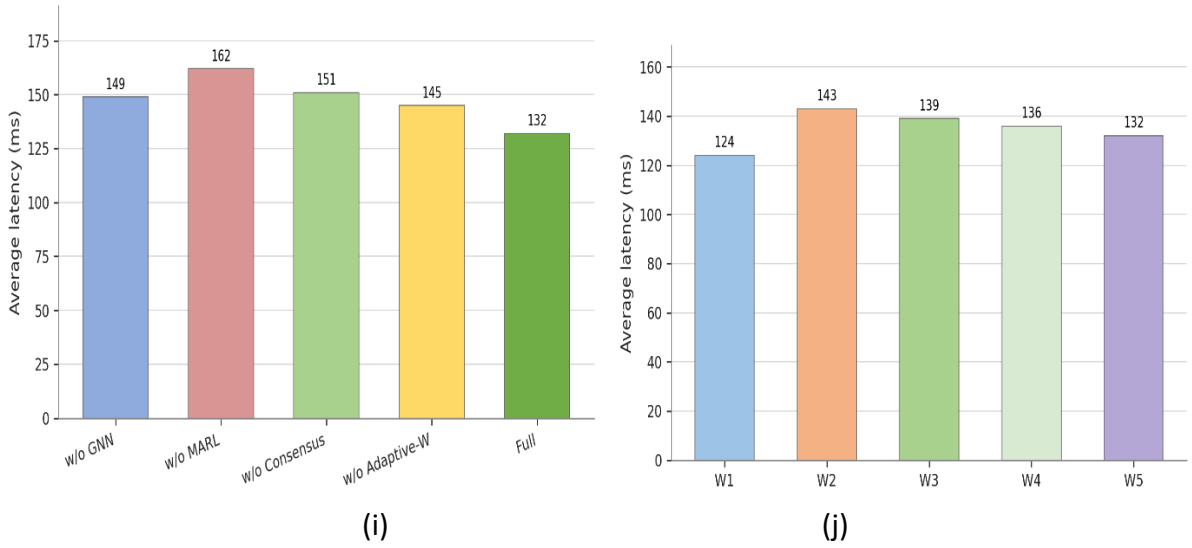


Fig. 2. (a) Average latency under high workload, (b) Total energy consumption comparison, (c) Deadline satisfaction ratio, (d) Load balancing using Jain fairness index, (e) Task success rate under fog-node failures, (f) Scalability with increasing task volume, (g) Communication overhead comparison, (h) Carbon emission comparison, (i) Ablation study, and (j) Weight Analysis

Conclusion and Future Work

- This paper introduced DAGMARL-Fog, a decentralized graph-aware multi-agent reinforcement learning framework for IoT–fog–cloud task scheduling.
- The framework integrates a graph representation of the state space, MARL for task assignment, feasibility filtering, consensus-based scheduling, and adaptive objective weighing.
- From the experimental results, we can see that DAGMARL-Fog attains 132 ms latency, 7900 J energy, 95.6% deadlines satisfaction rate, 0.94 of Jain fairness index, 85.0% task success ratio with 40% node failures, 186 ms latency when the number of tasks equals to 100,000, 265 KB communication overhead per each interval, and 750 gCO_2e carbon emission.
- It means that the graph-aware decentralized scheduling is efficient in the case of the dynamic fog environment.
- For further research, DAGMARL-Fog will be extended in terms of deployment on fog testbed, online adaptation to non-stationary workload, secure inter-fog coordination, private state sharing, carbon intensity estimation, and light policy compression for fog nodes.

References

1. H. Topcuoglu, S. Hariri, and M. Y. Wu, Performance-effective and low-complexity task scheduling for heterogeneous computing, IEEE Transactions on Parallel and Distributed Systems, vol. 13, no. 3, pp. 260–274, 2002.
2. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182–197, 2002.
3. V. Mnih et al., Human-level control through deep reinforcement learning, Nature, vol. 518, pp. 529–533, 2015.
4. J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, Proximal policy optimization algorithms, arXiv preprint arXiv:1707.06347, 2017.
5. R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, Multi-agent actor-critic for mixed cooperative-competitive environments, Advances in Neural Information Processing Systems, 2017.
6. T. N. Kipf and M. Welling, Semi-supervised classification with graph convolutional networks, International Conference on Learning Representations, 2017.
7. R. Mahmud, K. Ramamohanarao, and R. Buyya, Latency-aware application module management for fog computing environments, ACM Transactions on Internet Technology, vol. 19, no. 1, pp. 1–21, 2018.
8. C. Sonmez, A. Ozgovde, and C. Ersoy, EdgeCloudSim: An environment for performance evaluation of edge computing systems, Transactions on Emerging Telecommunications Technologies, vol. 29, no. 11, e3493, 2018.
9. C. Reiss, J. Wilkes, and J. L. Hellerstein, Google cluster-usage traces: Format + schema, Google Inc., 2011.
10. R. Jain, D. M. Chiu, and W. R. Hawe, A quantitative measure of fairness and discrimination for resource allocation in shared computer systems, DEC Research Report TR-301, 1984.
11. Chhabra, G.S., Satti, S.K., Rajareddy, G.N.V. et al. Time-and-Traffic-aware collaborative task offloading with service caching-replacement in cloud-assisted mobile edge computing. Cluster Comput 28, 900 (2025). https://doi.org/10.1007/s10586-025-05629-x
12. Chhabra, G. S., Rajareddy, G. N., Mahapatra, A., Mangalampalli, S. S., Sahoo, K. S., Sethi, D., & Mishra, K. (2025). Deep Learning-centric Task Offloading in IoT-Fog-Cloud Continuum: A State-of-the-Art Review, Open Research Issues and Future Directions. IEEE Access.