Graph Reinforcement Learning Under Visibility Constraints for Federated Vehicular Edge Computing

Ayoub ESSAFI, Christophe DE VAULX, Oussama HABACHI and Gérard Chalhoub
Université Clermont Auvergne, Clermont Auvergne INP, CNRS, LIMOS, Clermont-Ferrand, France

***Abstract*—Vehicular edge computing requires each Vehicular User Equipment (VUE) to decide, under tight latency and resource constraints, whether to execute AI and non-AI workloads locally, at a Mobile Edge Computing (MEC) server, or in the cloud. This paper studies whether privileged information available during training can be converted into a high-performing policy that remains deployable under restricted visibility. We formulate task offloading in federated vehicular edge-cloud systems as a hierarchical Stackelberg decision problem with deadline-aware MEC admission control, cross-operator federation, and heterogeneous graph representations of VUE, base-station, MEC, and cloud. A graph reinforcement learning teacher is trained with privileged visibility, including inter-vehicle context and cross-operator MEC state, and is then distilled into a legal-only student that never observes privileged inputs at inference. Realistic urban-mobility simulations show that the privileged policy outperforms a strong analytical price-based scheduler under high contention, while the deployable student remains competitive across MEC densities and provides clear gains under non-stationary traffic bursts.**

***Index Terms*—Vehicular edge computing, task offloading, graph neural networks, reinforcement learning, privileged information, policy distillation, Stackelberg games, MEC federation.**

I. Introduction

Connected and autonomous vehicles increasingly generate computation-intensive and latency-critical workloads, including object detection, anomaly detection, trajectory prediction, and cooperative perception. Executing such workloads locally is often infeasible because on-board processors are constrained by energy, thermal, and cost budgets. Vehicular Edge Computing (VEC) mitigates this limitation by allowing each Vehicular User Equipment (VUE) to execute tasks locally, offload them to a nearby Mobile Edge Computing (MEC) server, or forward them to the cloud under task-specific deadlines, energy budgets, and monetary constraints [1]. The resulting offloading problem is intrinsically a resource allocation problem: radio association changes with mobility, MEC reachability evolves over time, queues build under contention, and a decision that is optimal for one vehicle may become harmful when many neighboring vehicles make similar choices.

This contention-sensitive behavior makes task offloading difficult to solve with static or purely local rules [2]–[4]. When MEC capacity is scarce relative to demand, routing a task to a congested server may trigger a deadline miss, whereas local execution, cloud execution, or a less obvious federated MEC choice may avoid it. A deployable offloading policy must therefore adapt to time-varying topology, bursty demand, heterogeneous compute resources, AI-competence constraints, and cross-operator federation, while remaining lightweight enough to run repeatedly in a mobile edge system.

Analytical schedulers and deep reinforcement learning (DRL) have both been used for edge offloading [2], [5], [6]. However, their comparison often hides a fundamental asymmetry: the information available during decision-making is not the same for all controllers. Centralized schedulers, simulators, and analytical baselines may exploit global or semi-global state, such as the aggregate demand of vehicles attached to the same base station, the queue state of neighboring MEC servers, or the load of MECs reachable through federation agreements. A policy deployed on a VUE, by contrast, can only rely on information that is locally observable, legally shareable, or explicitly advertised by the infrastructure. This visibility gap raises a question that is rarely isolated in VEC offloading studies: *how much performance comes from the offloading objective itself, and how much comes from the network state that the controller is allowed to observe?*

This paper addresses that question through graph reinforcement learning for federated vehicular edge-cloud systems. We model the VUE–MEC–cloud interaction as a hierarchical Stackelberg decision process in which MEC operators act as leaders that advertise admission prices, while VUEs act as followers that select local, edge, cloud, or drop actions under deadline, budget, and energy constraints. The time-varying system state is represented as a heterogeneous graph whose nodes correspond to VUEs, base stations, MEC servers, and the cloud, and whose typed edges capture radio association, backhaul connectivity, and cross-operator MEC federation. This graph formulation is well suited to mobile edge systems because it naturally represents changing topology and supports policies whose parameters are independent of the number of vehicles or MEC servers.

MEC federation and edge support are addressed in ETSI and 3GPP specifications [7]–[9]. From a decision-theoretic perspective, such systems naturally induce hierarchical interactions: operators advertise prices or admission conditions, while users respond by selecting local, edge, cloud, or drop actions. Stackelberg and hierarchical games provide the leader–follower framework for this setting [10]. Our prior analytical scheduler, CAPTOE [11], follows this line by designing a decentralized competence-aware pricing and offloading mechanism under limited information. The present paper addresses a different question: rather than deriving a new analytical pricing rule, we investigate whether a learned graph policy can

979-8-3195-0167-7/26/$31.00 ©2026 IEEE

exploit richer training visibility, be distilled into the same deployable visibility regime as CAPTOE, and avoid per-instance recomputation at inference. The central methodological idea is to separate *training visibility* from *deployment visibility*. During training, a privileged graph policy is allowed to exploit information that a centralized simulator or infrastructure-side trainer can observe, including inter-vehicle context aggregated through base-station neighborhoods and cross-operator MEC state propagated through federation edges. At deployment, however, the VUE-side policy must operate under strict legal visibility, matching the information available to a decentralized analytical scheduler. We therefore train a privileged graph teacher and distill it into a legal-only student that never receives privileged inputs at inference.

II. System Model

We consider a federated vehicular edge-cloud network composed of a set $\mathcal{V}$ of VUEs, a set $\mathcal{B}$ of BSs, a set $\mathcal{M}$ of MEC servers owned by P operators, and a remote cloud server. Each VUE may generate computation tasks and can execute them locally, offload them to one of its accessible MEC servers, transmit them to the cloud, or drop them if no feasible execution option is available. The considered workloads include both AI and non-AI tasks. Representative AI workloads include real-time object detection from camera streams, trajectory prediction, cooperative perception, and anomaly detection. These tasks may require GPU-accelerated inference, CPU preprocessing, strict deadlines, and task-specific AI competences. Non-AI workloads include map matching, sensor fusion, route planning, and control-related computations, which are primarily CPU-intensive. Each task is represented by a tuple

$$\tau_i = \big(D_i, C_i^{\mathrm{CPU}}, C_i^{\mathrm{GPU}}, T_i^{\mathrm{DL}}, b_i, e_i, \chi_i\big), \tag{1}$$

where D_i is the input payload size, C_i^{CPU} and C_i^{GPU} are the required CPU and GPU cycles, T_i^{DL} is the completion deadline, b_i is the monetary budget, e_i is the available VUE energy budget, and χ_i denotes the required AI competence.

Each MEC server $m \in \mathcal{M}$ is characterized by heterogeneous computing capacities, including CPU and GPU processing rates $(f_m^{\mathrm{CPU}}, f_m^{\mathrm{GPU}})$, a queue state, an operator identity, a backhaul delay, and advertised resource prices. Due to memory limitations, model availability, licensing constraints, or operator policies, each MEC implements only a subset $\mathcal{C}_m$ of AI competences and can execute natively only the AI workloads whose required competence belongs to $\mathcal{C}_m$. If an AI task requiring competence χ_i is offloaded to a MEC such that $\chi_i \notin \mathcal{C}_m$, the task can still be processed through a degraded compatibility mode: the VUE performs local preprocessing requiring C_i^{pre} additional CPU cycles and transmits an inflated payload $\widetilde{D}_i > D_i$. MEC servers are interconnected through federation interfaces that allow task redirection across operators. Such redirection incurs a roaming surcharge and an additional backhaul delay. The cloud is assumed to provide abundant computing capacity and broad AI competence support, but it incurs a larger backhaul delay than MEC execution. For each task the serving policy selects $a \in \{\mathrm{local}, \mathrm{MEC}_1, \ldots, \mathrm{MEC}_K, \mathrm{cloud}, \mathrm{drop}\}$, where $\mathcal{M}_i \subseteq \mathcal{M}$ ($|\mathcal{M}_i| \le K$) are the MECs reachable by VUE i. A task is satisfied if completed before its deadline. For a task τ served by node m, the completion delay is

$$T_{i,m} = \underbrace{D_{i,m}^{\mathrm{up}}/R_{i,m}}_{\text{transmit}} + \underbrace{B_m}_{\text{backhaul}} + \underbrace{W_m}_{\text{queue}} + \underbrace{\frac{C^{\mathrm{CPU}}}{f_m^{\mathrm{CPU}}} + \frac{C^{\mathrm{GPU}}}{f_m^{\mathrm{GPU}}}}_{\text{execute}}, \tag{2}$$

where $R_{i,m}$ is the achievable rate, B_m the backhaul delay (0 for co-located MECs), and W_m the queueing delay.

We distinguish two visibility regimes. In the privileged regime, used only for teacher training, the policy observes the full heterogeneous graph, including reverse BS→VUE edges that expose the context of co-located vehicles and inter-MEC federation edges that propagate cross-operator MEC state. In the legal regime, used for deployment, the VUE-side policy observes only its own task and mobility state, its associated BS, and the descriptors and advertised prices of its reachable MEC candidates. A VUE consumes computation energy when executing a task locally and radio energy when offloading it. For task τ_i served by node m, the VUE-side energy cost is

$$E_{i,m} = \begin{cases} \xi_V^{\mathrm{cpu}}\, C^{\mathrm{CPU}} + \xi_V^{\mathrm{gpu}}\, C^{\mathrm{GPU}}, & \text{local}, \\ P_i^{\mathrm{tx}}\, \widetilde{D}/R_{i,m} + \xi_V^{\mathrm{cpu}}\, C^{\mathrm{pre}}, & \text{MEC}, \\ P_i^{\mathrm{tx}}\, D_i/R_{i,m}, & \text{cloud}, \end{cases} \tag{3}$$

where ξ_V^{cpu} and ξ_V^{gpu} are the VUE's per-cycle CPU and GPU energy coefficients, P_i^{tx} is the uplink transmit power, and $R_{i,m}$ is the uplink rate toward the selected execution node. Here $\widetilde{D}$ and C^{pre} capture AI-competence mismatch.

III. The Proposed Method

A. Stackelberg federation

The interaction between MEC operators and VUEs is modeled as a Stackelberg game. MEC operators act as *leaders*: each MEC m advertises CPU/GPU resource prices $\mathbf{p}_m = (p_m^{\mathrm{CPU}}, p_m^{\mathrm{GPU}})$ and applies deadline-aware admission control according to its current queue, available resources, supported AI competences, and federation agreements. VUEs act as *followers*. Given the posted prices $\mathbf{p} = \{p_m\}_{m\in\mathcal{M}}$, each VUE selects the action maximizing its utility $U_i(a; \mathbf{p})$, combining on-time completion reward, monetary cost, delay, and energy. Anticipating the aggregate follower response, the infrastructure updates prices on a slower time scale to maximize operator revenue from completed tasks.

B. Heterogeneous graph and encoder

At each decision step, the simulator state is encoded as a heterogeneous graph $G = (\mathcal{N}, \mathcal{E})$ with four node types. *VUE* nodes carry position, velocity, home-operator one-hot, remaining budget/energy, and the current task descriptor; *BS* nodes carry location and association load; each *MEC* node carries a feature vector (normalized CPU and GPU capacity and price, supported AI competences, operator one-hot, and backhaul); the *Cloud* node carries an analogous summary. Typed edges encode VUE→BS radio association, BS→MEC backhaul, and MEC↔MEC federation.

A Heterogeneous Graph Transformer (HGT) [12] with L type-aware multi-head attention layers and hidden width D maps G to per-node embeddings $\{h_v\}$. The legal and privileged graphs share this architecture and differ *only* in which edges and features are present, which is what makes the visibility comparison exact.

C. Hierarchical actor–critic

The *follower* (VUE) actor consumes the VUE embedding h_i, a 6-dimensional summary of each of its K candidate MECs, and task/cloud features, and outputs a categorical policy $\pi_\theta(a \mid s_i)$ over the $3{+}K$ actions $\{\text{local}, \text{cloud}, \text{drop}, \text{MEC}_1, \ldots, \text{MEC}_K\}$. The *leader* (MEC) actor emits admission prices on a slower time scale, realizing the Stackelberg hierarchy. A *centralized* critic over the graph embedding estimates the reward value V and a vector of per-constraint cost values. The follower reward decomposes as

$$r_i = \underbrace{\rho \mathbb{1}[T_i \le T_i^{\mathrm{DL}}]}_{\text{on-time revenue}} - \underbrace{m_i}_{\text{money}} + \underbrace{\gamma_{\mathrm{qos}}\, \phi(T_i, T_i^{\mathrm{DL}})}_{\text{QoS shaping}}, \tag{4}$$

where $\phi(\cdot)$ is a smooth deadline-slack term and γ_{qos} weights it. Optimization uses multi-agent PPO [13] with a clipped surrogate and generalized advantage estimation [14].

D. Constrained objective

With per-decision costs c_d (deadline-miss), c_b (budget overrun), c_e (energy) and thresholds $\tau_\bullet$, the policy solves the constrained MDP via the Lagrangian with dual ascent on the multipliers $\lambda_k \ge 0$ [15]:

$$\max_{\pi}\ \mathbb{E}[\textstyle\sum_t r_t] - \sum_{k \in \{d,b,e\}} \lambda_k \big(\mathbb{E}[c_k] - \tau_k\big). \tag{5}$$

E. Privileged-to-deployable distillation

A *teacher* π_T is trained at privileged visibility under (5). A *student* π_S of identical architecture is constrained to legal visibility at *every* training step, it never observes a privileged input, and imitates the teacher on the teacher's realized states. For each active decision, with teacher/student logits z_T, z_S, temperature κ and hard-anchor weight β, we minimize

$$\mathcal{L} = \kappa^2\, \mathrm{KL}\big(\sigma(z_T/\kappa) \,\|\, \sigma(z_S/\kappa)\big) + \beta\, \mathrm{CE}\big(z_S,\, \arg\max z_T\big), \tag{6}$$

the Learning-by-Cheating recipe [16]. Because π_S is legal-only *by construction*, there is no train/test visibility mismatch: the deployed artifact is exactly the evaluated policy. Algorithm 1 summarizes the procedure.

Algorithm 1 Privileged-to-deployable training

```
Train teacher π_T on the privileged graph G^priv via constrained MARL (5)
Initialize legal-only student π_S (identical architecture)
for each distillation iteration do
    Roll out π_T; at each step build both G^priv and the legal graph G^legal
    for each active VUE decision on the realized state do
        z_T ← π_T(G^priv);  z_S ← π_S(G^legal)
        accumulate L (6); execute teacher action (on-policy)
    end for
    update π_S by ∇_{θ_S} L
end for
deploy π_S on G^legal only
```

IV. Simulation Results

We simulate the road network of Lyon, France, with $B = 50$ base stations, n MEC servers owned by P operators, and a cloud. Unless noted, $N = 2000$ VUEs move for 5000 steps; vehicles move and budgets replenish every step. The principal baseline, CAPTOE [11], is our prior analytical price-based scheduler at legal visibility. We additionally report MIN_DELAY, a greedy per-task delay-minimizing oracle with access to exact queue-wait state, as a centralized analytical upper reference. DRL_MEC is a fixed-dimension DRL policy and LOCAL_ONLY executes every task on the VUE.

A. Privileged performance and the contention regime

Table I reports the privileged policy against other methods. The privileged policy exceeds CAPTOE under high contention and is competitive as contention relaxes.

When capacity is scarce (n=5), routing to a server that neighbors are also about to choose is the dominant failure mode, and this anticipatory signal is what allows the privileged policy to avoid it. When capacity is abundant (n=25), almost any non-degenerate choice meets the deadline, so the same signal is redundant and every method saturates which is why the privileged margin, and even the gap with the MIN_DELAY oracle, shrink to noise as n grows.

TABLE I
PRIVILEGED GRL VS. BASELINE DEADLINE SATISFACTION (%).

n MECs	Privileged GRL	CAPTOE	MIN_DELAY	DRL_MEC
5	**95.40 ±0.35**	94.43	96.43	88.83
15	97.57 ±0.36	97.68	97.86	91.82
25	98.23 ±0.19	98.30	98.54	93.34

B. The cost of visibility

The privileged policy is not deployable: its follower consumes inter-vehicle edges a VUE cannot observe. *Forcing the privileged follower to legal visibility at inference*, a naive train/test restriction, collapses it to ≈93.3, *below* the baseline, exposing a ∼2.1 pp train/test mismatch. The privileged margin is the cross-entity graph structure the follower sees during

training. Our deployable student which is trained legal-only from the start avoids this mismatch: it recovers to ≈ 94.4, *above* the naively-restricted policy and *competitive with* CAPTOE across all scales, as shown in Figure 1. This is the central deployability result: privileged knowledge in training, no privileged input at deployment.

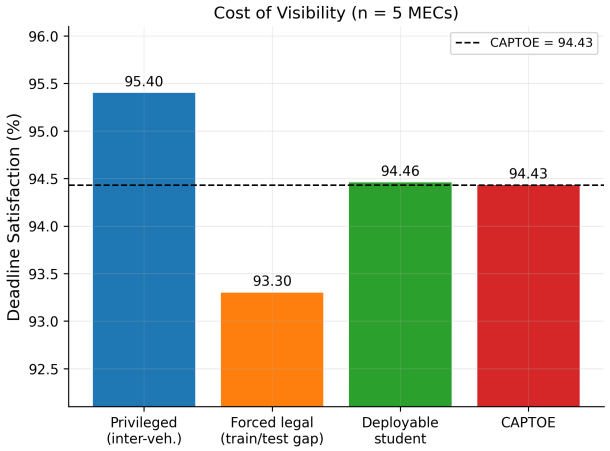


Fig. 1. Cost of visibility at n=5.

C. Generalization across network scale

Figure 2 shows that deadline satisfaction rises smoothly and monotonically across both seen and unseen densities, with no irregularity at interpolated or extrapolated scales.

Figure 3 shows graceful degradation under *demand*: for 4500 vehicles (2.25× the 2000-VUE training point), deadline satisfaction declines smoothly with no cliff, and steeply only under the tightest capacity (n=5), where it remains above 88%. We train the student only on $n \in \{5, 15, 25\}$ and evaluate it zero-shot at the unseen densities $n \in \{10, 20, 30, 35\}$. Under bursty arrivals (eval-only, no retraining), the student additionally exceeds CAPTOE, as CAPTOE's damped price update cannot re-price fast enough when arrivals jump.

V. Conclusion

We proposed federated VEC offloading as GRL under explicit visibility constraints. A privileged graph policy exceeds a strong analytical scheduler under high contention, and a distilled legal-only student deploys without privileged input while remaining competitive across scales and outperforming the visibility-restricted teacher.

References

[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, 2017.

[2] L. Zhao *et al.*, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, 2019.

[3] M. Hossain *et al.*, "Dynamic task offloading for cloud-assisted vehicular edge computing networks: A non-cooperative game theoretic approach," *Sensors*, vol. 22, 2022.

[4] F. Dai, G. Liu, Q. Mo *et al.*, "Task offloading for vehicular edge computing with edge-cloud cooperation," *World Wide Web*, vol. 25, pp. 1999–2017, 2022.

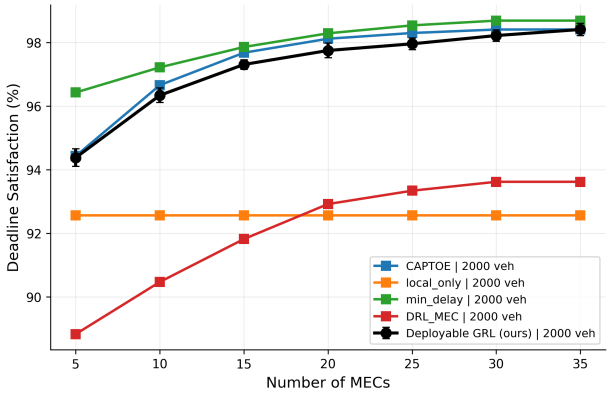


Fig. 2. Deadline satisfaction across MEC density (2000 VUEs, 5 seeds).

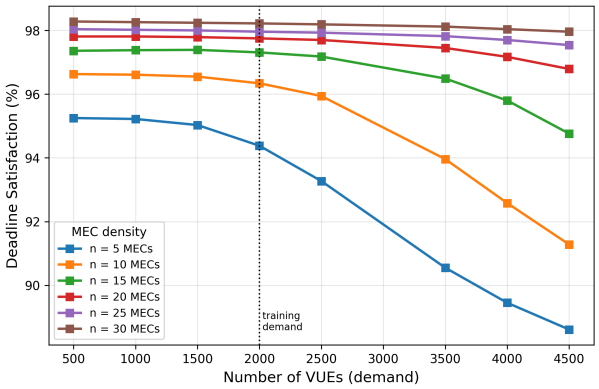


Fig. 3. Demand robustness: deadline satisfaction vs. vehicle population at each MEC density.

[5] M. Tang and V. W. S. Wong, "Deep reinforcement learning for task offloading in mobile edge computing systems," *IEEE Transactions on Mobile Computing*, vol. 21, no. 6, pp. 1985–1997, 2022.

[6] M. Alsahfi *et al.*, "Proximal policy optimization for vehicular big data offloading across edge, regional, and cloud layers," *Journal of Grid Computing*, 2025.

[7] "ETSI GS MEC 040: Multi-access edge computing (mec); federation enablement apis," ETSI, Tech. Rep., 2023.

[8] "ETSI GS MEC 002: Multi-access edge computing (mec); phase 2: Use cases and requirements," ETSI, Tech. Rep., 2023.

[9] "3GPP TR 23.748 V17.0.0: Study on enhancement of support for edge computing in 5g core network (5gc)," 3GPP, Tech. Rep., 2020.

[10] T. Başar and G. J. Olsder, *Dynamic Noncooperative Game Theory*, 2nd ed. Philadelphia, PA: SIAM, 1999.

[11] A. Essafi, O. Habachi, and C. De Vaulx, "Decentralized competence-aware task offloading for federated multi-access edge computing," in *Proc. IEEE Int. Wireless Communications and Mobile Computing Conf. (IWCMC)*, 2026, to appear.

[12] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "Heterogeneous graph transformer," in *Proceedings of The Web Conference (WWW)*, 2020.

[13] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of PPO in cooperative multi-agent games," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[14] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *International Conference on Learning Representations (ICLR)*, 2016.

[15] E. Altman, *Constrained Markov Decision Processes*. Chapman & Hall/CRC, 1999.

[16] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by cheating," in *Conference on Robot Learning (CoRL)*, 2019.