Stability-Aware Multi-Agent Reinforcement Learning for Dynamic Resource Management in Vehicular Edge Networks

Ali Bouriga
Univ Toulouse, Toulouse INP
CNRS, IRIT
Toulouse, France
ali.bouriga@irit.fr

Rahim Kacimi
Univ Toulouse, Toulouse INP
CNRS, IRIT
Toulouse, France
rahim.kacimi@irit.fr

Riadh Dhaou
Univ Toulouse, Toulouse INP
CNRS, IRIT
Toulouse, France
riadh.dhaou@irit.fr

***Abstract*—Vehicular Edge Computing must meet strict deadlines under dynamic traffic while limiting energy use. We propose MA-LDQN, a multi-agent Lyapunov-guided Deep Q-Network in which each Roadside Unit independently trains and executes a DQN agent using only local observations (task features, resources, virtual-queue backlog and neighbor loads). A Lyapunov-inspired reward balances deadline success with queue stability. SUMO simulations show hit rates of 72.2% (1× load) and 46.5% (2× load) together with substantially lower energy and more selective offloading than aggressive baselines, delivering a strong practical trade-off without centralized coordination.**

***Index Terms*—Vehicular networks, Vehicular Edge Computing, Multi-Agent Reinforcement Learning, Intelligent Transportation Systems**

I. Introduction and Related Work

The rapid proliferation of onboard sensing in modern vehicles has fundamentally altered the demands placed on road infrastructure. Today's cars run applications ranging from infotainment to safety-critical systems such as ADAS and autonomous driving [1], [2], generating continuous sensor data streams that require real-time processing under tight latency, energy, and reliability constraints — well beyond what onboard hardware alone can handle [3].

Several offloading paradigms have been explored. Vehicular Fog Computing (VFC) [4] distributes tasks among nearby vehicles but raises trust and privacy concerns. Cloud computing offers scalable capacity yet suffers from high latency and congestion in mobile environments. Vehicular Edge Computing (VEC) [5] strikes a better balance by placing compute resources at Roadside Units (RSUs) close to the data source, reducing end-to-end latency and core network dependence. VEC remains challenging however: resources are constrained, shared across dynamic vehicle populations, and must meet heterogeneous real-time deadlines under unpredictable mobility and wireless channel conditions.

Existing solutions mostly rely on single-agent DRL or centralized optimization, with few enabling direct RSU collaboration. Huang et al. [6] proposed CORA, a Twin Delayed DDPG framework that outperforms classical heuristics but assumes at most one task per vehicle per slot, same-slot completion, and permanent RSU coverage — assumptions that rarely hold in practice. Xie et al. [7] introduced a Transformer-based DRL approach with adaptive state and action spaces, though it is restricted to a single edge server. Siddharth et al. [8] proposed HAPTA, a UCB-based horizontal offloading algorithm across RSUs, but it relies on cloud-mediated coordination and assumes single-core RSU processing. These limitations — centralized control, simplified mobility, and lack of queue stability consideration — motivate a more realistic, decentralized approach.

We propose *MA-LDQN* (Multi-Agent Lyapunov-guided Deep Q-Network), a distributed reinforcement learning framework for stable task offloading in VEC. Each RSU runs an independent DQN agent whose state incorporates a Lyapunov virtual queue signal and whose reward embeds a drift-plus-penalty term to jointly optimize deadline satisfaction and congestion control. Agents coordinate implicitly through backhaul-exchanged queue-state information, with no centralized controller. The main contributions are:

- A decentralized multi-agent architecture where neighboring RSUs independently learn offloading policies via deep Q-networks, enabling collaborative edge resource sharing without central coordination.
- A Lyapunov-driven reward and state design integrating queue backpressure into DQN learning, jointly accounting for mobility, channel quality, deadlines, and congestion.
- A comprehensive evaluation on a realistic SUMO-based urban testbed showing MA-LDQN outperforms established heuristics and approaches an ORACLE solver.

The remainder is organised as follows: Section II describes the system model; Section III formulates the problem and presents MA-LDQN; Section IV reports and analyses results; Section V concludes.

II. System Model

A. Network and Mobility Scenario

We simulate an urban VEC environment based on Toulouse, France, using a 5×4 km road network extracted

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

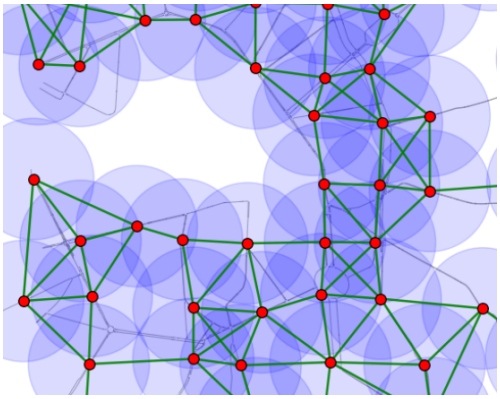


Fig. 1: RSU placement across the Toulouse road network.

TABLE I: MCS Table: SNR Thresholds and Data Rates

MCS	1	2	3	4	5	6	7	8
Min. SNR (dB)	10	13	16	19	22	25	28	30
Rate (Mbps)	3.0	4.5	6.0	9.0	12.0	18.0	24.0	27.0

from OpenStreetMap and imported into SUMO. One hundred RSUs are distributed via a grid-based scheme anchored to road segments (Fig. 1), yielding higher density near busy intersections. Each RSU covers a 500 m radius for wireless access and edge compute, per ETSI ITS guidelines. SUMO's microscopic traffic engine generates realistic vehicle dynamics; density grows progressively over each 1600-second episode to around 350 vehicles, inducing realistic congestion patterns. Seeds are varied across runs for scenario diversity.

B. Communication and Backhaul Model

Vehicles associate with their nearest RSU over an IEEE 802.11p/ETSI ITS-G5 link [9], modelled via log-distance path loss ($P_{\text{tx}} = 23$ dBm, $N_0 = -88$ dBm, $\text{PL}_0 = 40$ dB, $\eta = 3.2$). A link is viable when $d \le 500$ m and $\text{SNR}_{\text{dB}} \ge 10$ dB. Uplink throughput follows adaptive MCS (Table I), giving latency $\tau_i^{\text{ul}} = 8s_i/R(\text{SNR}_{\text{dB}})$ deducted from the task's deadline budget. At completion, downlink feasibility is verified against the vehicle's projected position; delivery fails if the projected distance exceeds 500 m, downlink SNR falls below 8 dB, or a shadowing event occurs ($p_{\text{shadow}} = 0.04$) — counting as a miss even if compute finished on time. Offloaded tasks incur a fixed backhaul delay $\tau^{\text{bh}} = 30$ ms, giving end-to-end latency:

$$\tau_i^{\text{e2e}} = \tau_i^{\text{ul}} + \tau^{\text{bh}} + w_{r_n}(t) + c_i \le D_i \tag{1}$$

C. Task Generation Model

Each vehicle alternates between idle and active burst states. Activation occurs at each 0.1 s tick with probability $p_{\text{on}} = \min(\alpha\rho, 0.99)$ ($\alpha = 0.10$, $\rho \in \{1.0, 2.0\}$); active vehicles revert with $p_{\text{off}} = 0.04$ (mean burst 2.5 s). During bursts, tasks arrive at rate:

$$\lambda_v(t) = \min(\lambda_0\rho + \kappa\, v_{\text{speed}}\, \phi(t),\ \lambda_{\max}\rho) \tag{2}$$

TABLE II: Task Type Parameters

Type	s_i (MB)	c_i (s)	D_i (s)	Priority
ADAS	2.0	0.15	0.25	3
Sensor Fusion	12.0	0.35	0.50	2
Navigation	0.5	0.05	1.00	1
Infotainment	2.0	0.20	2.00	0

where $\lambda_0 = 6.5$ tasks/s, κ scales with speed, $\phi(t) \in \{1.0, 1.4\}$ is a rush-hour factor, and $\lambda_{\max}$ caps the rate. Task types are sampled from a speed-dependent distribution over four classes (Table II); faster vehicles favour latency-sensitive types. Tasks are served under EDF, expire upon deadline violation, and those in the final 2.0 s are excluded from metrics.

D. Edge Computing Resource Model

Each RSU runs as a $K = 8$-core server linked to up to six neighbours via wired backhaul. Tasks execute non-preemptively with service time $\tilde{c}_i = c_i \cdot \mu$ ($\mu = 1.6$). EDF scheduling [10] governs dispatch: expired tasks are purged each tick, then the tightest-deadline task is sent to the first free core. A real-time load score estimates expected wait for a new arrival:

$$\ell_r(t) = \begin{cases} 0 & |\mathcal{Q}_r| = 0,\ K_{\text{free}} > 0 \\ \min_k c_r^{(k)}(t) + \frac{|\mathcal{Q}_r|\bar{c}}{K_{\text{free}}} & \text{any core active} \\ \frac{|\mathcal{Q}_r|\bar{c}}{K} & \text{otherwise} \end{cases} \tag{3}$$

Offloading candidates are restricted to backhaul neighbours within the vehicle's wireless range, ensuring downlink reachability at any selected target.

III. Problem Formulation and Proposed Method

A. Problem Formulation

Let $\mathcal{I}$ be the set of tasks generated over $[0, T]$ with $T = 1{,}600$ s, excluding those arriving after $T' = 1{,}598$ s. Each task i must be routed immediately upon arrival to a target RSU within its reachable set $\mathcal{R}_i$. Offloading beyond the home RSU r_h^i adds $\tau^{\text{bh}} = 30$ ms, yielding:

$$L_i = \tau_i^{\text{ul}} + \tau^{\text{bh}} \cdot \mathbf{1}[a_i \ne r_h^i] + w_i + \tilde{c}_i \tag{4}$$

A task counts as a *hit* only when both the deadline and downlink conditions are satisfied:

$$\begin{aligned} H_i = {} & \mathbf{1}[L_i \le D_i] \cdot \mathbf{1}[B_i = 0] \\ & \cdot \mathbf{1}\left[\text{SNR}_{\text{dB}}\left(d\left(\mathbf{p}_{v_i}(t_{\text{arr}}^i + L_i), \mathbf{p}_{a_i}\right)\right) \ge \gamma_{\min}^{\text{dl}}\right] \end{aligned} \tag{5}$$

where $B_i \sim \text{Bernoulli}(0.04)$ models link shadowing. The objective is to maximise the overall hit rate:

$$\max_{\pi}\ \eta(\pi) = \frac{1}{|\mathcal{I}|}\sum_{i \in \mathcal{I}} H_i \tag{6}$$

All routing decisions must be causal, based solely on information available at t_{arr}^i: $\mathcal{O}_i = \{\mathbf{c}_r, \mathcal{Q}_r, \ell_r\}_{r \in \mathcal{R}_i} \cup \{s_i, D_i, \text{SNR}_i, \mathbf{p}_{v_i}\}$. This naturally frames the problem as a

Dec-POMDP [11], where agents share the cooperative objective $\eta(\pi)$ but observe only local neighbourhoods. With 100 RSUs each choosing among 7 actions, the centralised joint action space reaches 7^{100}, making exact solution intractable [12] and motivating the independent learning approach below.

B. Proposed Method: MA-LDQN

We propose *MA-LDQN* (Multi-Agent Lyapunov-guided Deep Q-Network), in which each RSU runs an independent DQN agent that learns an offloading policy from local observations. Lyapunov drift-plus-penalty (DPP) theory [13] is embedded directly into the state representation and reward signal to promote queue stability, without requiring a centralized critic or inter-agent gradient sharing.

State Representation: At each task arrival, the agent at RSU r constructs a 12-dimensional observation vector. Three features describe the incoming task: normalised size $\bar{s}_i$, effective remaining deadline $\bar{d}_i^{\text{eff}} = D_i - \tau_i^{\text{ul}}$, and uplink SNR $\bar{\gamma}_i$. Three features capture the local RSU state: queue occupancy $\bar{q}_r$, core utilisation u_r, and the normalised Lyapunov virtual queue $\hat{Q}_r$, where Q_r evolves as:

$$Q_r(t+1) = \max\big(Q_r(t) - C_r,\, 0\big) + A_r(t) \tag{7}$$

with $A_r(t)$ the task arrival count and C_r a fixed drain rate proportional to RSU capacity. $\hat{Q}_r$ gives the agent a direct backpressure signal grounded in Lyapunov theory. The remaining six dimensions encode the load scores of up to six backhaul neighbours, zero-padded for unused slots. Unlike the more complex MAPPO variant, no temporal trend features are included, keeping the state compact and compatible with tabular-style Q-learning dynamics.

Action Space and Masking: Each agent chooses among 7 discrete actions: serve the task locally (action 0) or forward it to one of up to six reachable neighbours (actions 1–6). Neighbours outside wireless range or below the SNR threshold are masked out by assigning $-\infty$ to their Q-values before the $\arg\max$, preventing infeasible selections.

Lyapunov-Inspired Reward: The reward signal integrates task outcome with a queue stability penalty:

$$r_i = \xi_i - \alpha\, \hat{Q}_r \cdot \max\big(0,\, \ell_{\text{eff}} - \beta\big) - \rho_{\text{off}} \tag{8}$$

where $\xi_i \in \{1.8, -1.6, -0.4\}$ reflects task hit, expiry, or in-progress outcome, ℓ_{eff} is the post-backhaul RSU load, and ρ_{off} penalises low-gain migration decisions. The Lyapunov penalty activates only above a load threshold $\beta = 0.8$, avoiding unnecessary penalisation under moderate conditions; $\alpha = 0.7$ was set empirically. Rewards are clipped to $[-4, 4]$ for training stability. Note that unlike MAPPO, no deadline urgency weight u_i is applied, keeping the reward simpler and more aligned with standard DQN practice.

Training Procedure: Each RSU maintains an independent DQN with a two-layer MLP (64 units per layer, ReLU), trained via experience replay (buffer size 30,000) and a hard target network updated every 500 steps. Exploration follows ε-greedy decay from 1.0 to 0.05. Training runs for 60 episodes at $1\times$ load followed by 60 episodes at $2\times$ load, with no behavioural cloning warm-up and no curriculum beyond this two-phase schedule. Adam optimiser is used with learning rate 5×10^{-4} and batch size 256. A single trained policy is evaluated at both load regimes without any retraining.

IV. Results Discussion

We evaluate the MA-LDQN against five baselines upper bound under moderate ($1\times$, $\approx$1.75M tasks) and high ($2\times$, $\approx$3.87M tasks) load, using a *single trained policy* at both regimes without retraining. Baselines include **Local** (always local), **RANDOM** (uniform random routing), **Greedy** (offload to least-loaded neighbour), **Oracle** (exact neighborhood queue visibility, smallest estimated completion time), and **DQN** (independent Q-network per RSU).

Overall performance. As shown in Fig. 2, MA-LDQN attains a deadline hit rate of 72.2% at $1\times$ load and 46.5% at $2\times$ load. It clearly outperforms independent DQN (68.8% / 45.5%) and stays competitive with both Greedy (73.4% / 47.1%) and the Oracle (75.8% / 48.4%).

Offloading Behavior and Energy Consumption. Figures 3 and 4 highlight a core strength of MA-LDQN: competitive deadline performance with highly selective offloading. At $1\times$ load it offloads only 35.7% of tasks—well below Oracle (50.7%) and Greedy (76.8%)—yet matches or exceeds their hit rates. This selectivity stems directly from the Lyapunov term, which triggers offloading only when virtual-queue backpressure and load jointly signal a clear benefit. Under $2\times$ load the pattern is even more pronounced: MA-LDQN's offload rate falls to 31.0%, while every other offloading policy increases its rate. The Lyapunov penalty appears to discourage sending tasks to already-congested neighbors, favoring local execution when no advantageous target exists. Energy consumption mirrors this restraint. MA-LDQN uses 1.21 J and 1.24 J per task at the two loads—the lowest among offloading schemes and only modestly above pure local processing (0.90 J / 0.96 J). Greedy, by contrast, consumes 1.59 J and 1.73 J (31% higher at $1\times$), while the plain DQN baseline still requires 1.45 J and 1.57 J. The gap confirms that the Lyapunov reward, rather than the network architecture itself, is the main driver of the observed efficiency.

V. Conclusion

We presented a multi-agent Lyapunov-guided Deep Q-Network (MA-LDQN) for task offloading in Vehicular Edge Computing. Each roadside unit maintains an independent DQN agent trained solely on its local experiences, using a state that includes task features, local resource status, its own virtual-queue backlog, and neighbor load scores. Embedding Lyapunov queue-stability signals in the reward yields congestion-aware, deadline-sensitive decisions that outperform online baselines in both hit rate and energy efficiency under moderate and high loads. The fully decentralized design shows that stability-aware independent learning can operate effectively without global coordination in large-scale VEC

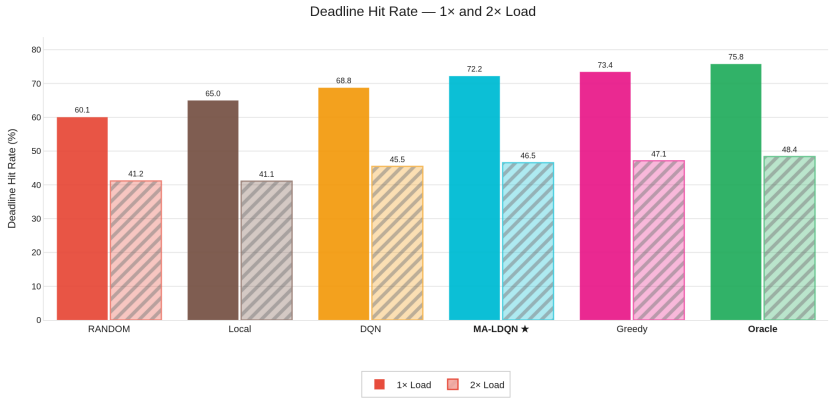


Fig. 2: Estimated energy consumption per task under $1\times$ and $2\times$ load.

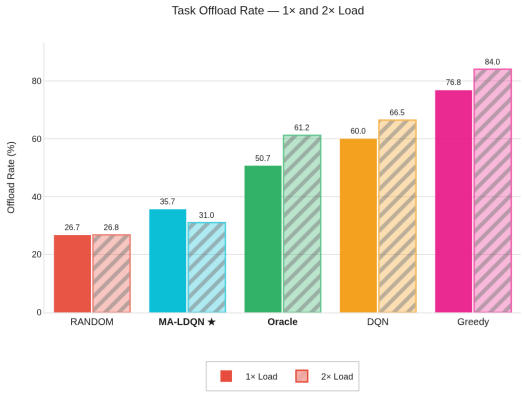


Fig. 3: Estimated energy consumption per task under $1\times$ and $2\times$ load.

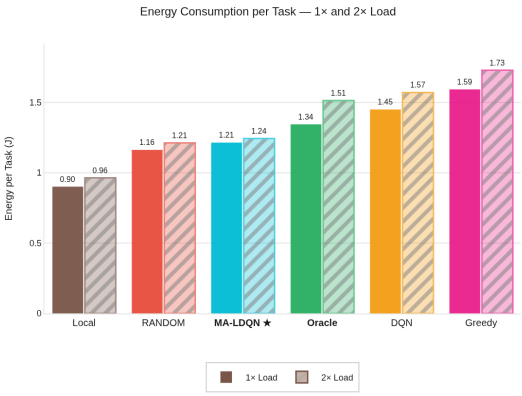


Fig. 4: Estimated energy consumption per task under $1\times$ and $2\times$ load.

environments. Future work will explore selective inter-RSU load signaling at inference, continual learning for evolving traffic conditions, and extension to multi-tier edge–cloud architectures.

REFERENCES

[1] P. V. Teixeira, D. Raposo, R. Lopes, and S. Sargento, "Software defined vehicles for development of deterministic services," *arXiv preprint arXiv:2407.17287*, 2024.

[2] A. Bouriga, R. Kacimi, and R. Dhaou, "Deep reinforcement learning-powered extended vehicular cooperative perception," in *2026 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE, 2026, pp. 802–807.

[3] L. Basset, A. Bouriga, and R. Dhaou, "Learning to talk: Emergent cam/cpm scheduling via mappo in constrained v2x networks," in *2025 International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*. IEEE, 2025, pp. 767–770.

[4] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *IEEE transactions on vehicular technology*, vol. 65, no. 6, pp. 3860–3873, 2016.

[5] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. Zhang, "Vehicular edge computing and networking: A survey," *Mobile networks and applications*, vol. 26, no. 3, pp. 1145–1168, 2021.

[6] J. Huang, J. Wan, B. Lv, Q. Ye, and Y. Chen, "Joint computation offloading and resource allocation for edge-cloud collaboration in internet of vehicles via deep reinforcement learning," *IEEE Systems Journal*, vol. 17, no. 2, pp. 2500–2511, 2023.

[7] Y. Xie, F. Zhang, Y. Fu, C. Xu, and T. Q. Quek, "Towards task number adaptive offloading in mec systems: A transformer-based drl approach," in *2025 IEEE 101st Vehicular Technology Conference (VTC2025-Spring)*. IEEE, 2025, pp. 1–5.

[8] T. A. Siddharth and D. Gangadharan, "Hapta: Adaptive learning based horizontal computation offloading over rsus in vec," in *2025 IEEE 101st Vehicular Technology Conference (VTC2025-Spring)*. IEEE, 2025, pp. 1–7.

[9] ETSI, "Intelligent transport systems (its); its-g5 access layer specification for intelligent transport systems operating in the 5 ghz frequency band," European Telecommunications Standards Institute, Tech. Rep. EN 302 663 V1.3.1, January 2020. [Online]. Available: https://www.etsi.org/deliver/etsi_en/302600_302699/302663/01.03.01_30/en_302663v010301v.pdf

[10] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM (JACM)*, vol. 20, no. 1, pp. 46–61, 1973.

[11] F. A. Oliehoek, C. Amato *et al.*, *A concise introduction to decentralized POMDPs*. Springer, 2016, vol. 1.

[12] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein, "The complexity of decentralized control of markov decision processes," *Mathematics of operations research*, vol. 27, no. 4, pp. 819–840, 2002.

[13] M. Neely, *Stochastic network optimization with application to communication and queueing systems*. Morgan & Claypool Publishers, 2010.