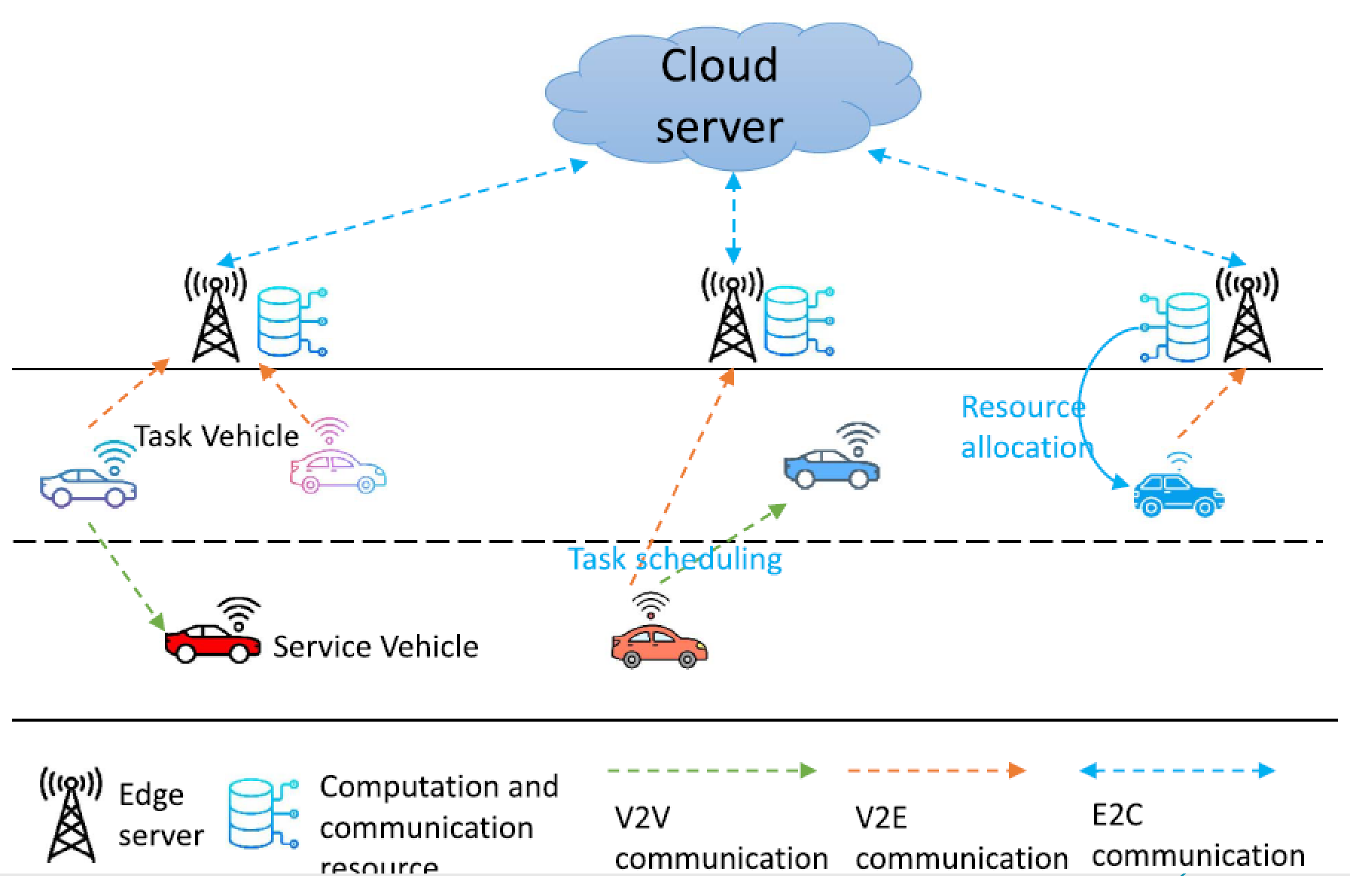
EPtask: Deep Reinforcement Learning Based Energy-E fficientand Priority-Aware Task Scheduling for Dynamic Vehicular Edge Computing

Mahan Bayhaghi



Cloud server
Task Vehicle
Resource allocation
Task scheduling
Service Vehicle
Edge server
Computation and communication resource
V2V communication
V2E communication
E2C communication

VEC

EPTask on VEC: PPO vs. DDPG/SAC & Baselines

- **Goal:** Priority-aware, energy-efficient task scheduling in **Vehicular Edge Computing (VEC)**
- **What we built:** A paper-aligned simulator + RL scheduler (PPO) with suggested baselines (SAC, DDPG, heuristic, local-only, offloading-only, random)
- **Key metrics:** deadline miss rate, completion time, mean energy, throughput.
- **Takeaway:** PPO learns smarter offloading & prioritization under mobility, deadlines, and energy limits.

Problem & Project Objectives

- **Challenge:** Where to compute each task (local/V2V/edge/cloud) and at what priority, under deadlines, mobility, and limited bandwidth/energy
- **Project objectives:**
  - Implement EPTask-style environment
  - Train PPO and compare against DDPG, SAC, heuristic, local-only, offloading-only, random
  - Report miss rate, completion time, energy, throughput
  - show scalability vs. #vehicles
- **Design constraints (per proposal & our choices):**
  - Non-preemptive + EDF
  - Distance-based TX power by default (learned-power kept for ablations).

Paper Alignment & Assumptions

- **Table III mapping (core params):**
    - **Radio:** V2I 100 MHz; noise ≈ −95 dBm (from −174 dBm/Hz over 100 MHz); vehicle TX ≈ 1 dBm (bins for tiers).
    - **Compute:** vehicle ≈ 1 GHz, edge ≈ 2 GHz, cloud ≈ 5 GHz (MIPS equivalents).
    - **Tasks:** size 2–20 Mb, cycles 2–20 × 10^9, 4 priority levels; **Poisson** arrivals (bursty).
- **Mobility & links:** 1-D movement; distance→SNR→rate
- **Scheduling rule: EDF**
- **Energy model:** TX + compute energy per task

TABLE III
EXPERIMENTAL PARAMETERS

Parameters	Value
Number of vehicles	10-50
Number of edge servers	8
Number of tasks	0-10
Required CPU cycles of the tasks C	2-20 cpu cycles
Computation power of edge servers	2 cpu cycles/s
Computation power of vehicles	1 cpu cycles/s
data size S	2-20 Mb
Bandwidth of edge server	100 MHz
Speed of vehicles	$\approx$ 25 m/s
Transmit power of vehicles	1 dBm
Execution power of vehicles	3-4 dBm
White Gaussian noise	-174 dBm/Hz
Power consumption coefficient ξ	10^{-11}
Power consumption coefficient γ	2
Received Signal Strength Indicator $RSSI$	-65 dBm
Transmit antenna gain TX	20 dBi
Receive antenna gain RX	-8 dBi
Signal attenuation SA	7 dB
Working frequency f	5GHz
Learning rate	0.0003
Size of Mini-batch	32
Number of steps in each episode	2048
Entropy loss coefficient	0.01

Simulator Architecture (What’s in Code)

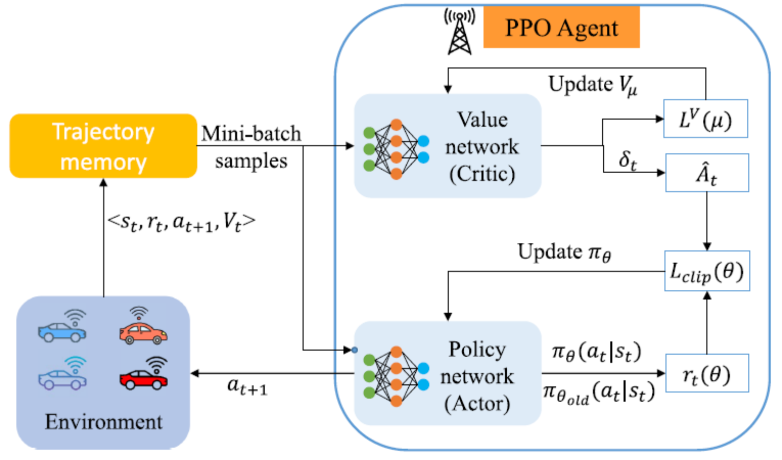
- **Env core (env/vec_env.py):**
  - Builds **Dict observation** (global/task/target/vehicle tensors)
  - **Action** per step on top-K tasks: *(offload target, priority)*
  - **Task generation:** Poisson arrivals; sample_tasks() from env/generators.py
  - **Link budget:** distance→SNR→rate
  - **Energy/time:** env/models.py (tx/compute time + energy functions)
  - **Reward:** −(z-time + z-energy) − λ·misses

- **Metrics & tools:**
  •env/metrics.py, scripts/eval_metrics.py, scripts/plot_metrics.py.
  •Grid & scalability: scripts/run_grid.py, scripts/plot_vs_vehicles.py.

RL Formulation & Methods

- **Action (per top-K tasks):**
  - **Offload target** ∈ {local, vehicles, edges, cloud}
  - **Priority level** ∈ {1..4}
  - *(default is distance-based, no power head)*
- **Reward:**
  - (λ_1·**z(completion time)** + λ_2·**z(energy)**) − λ_3·**#deadline_misses** (per step)

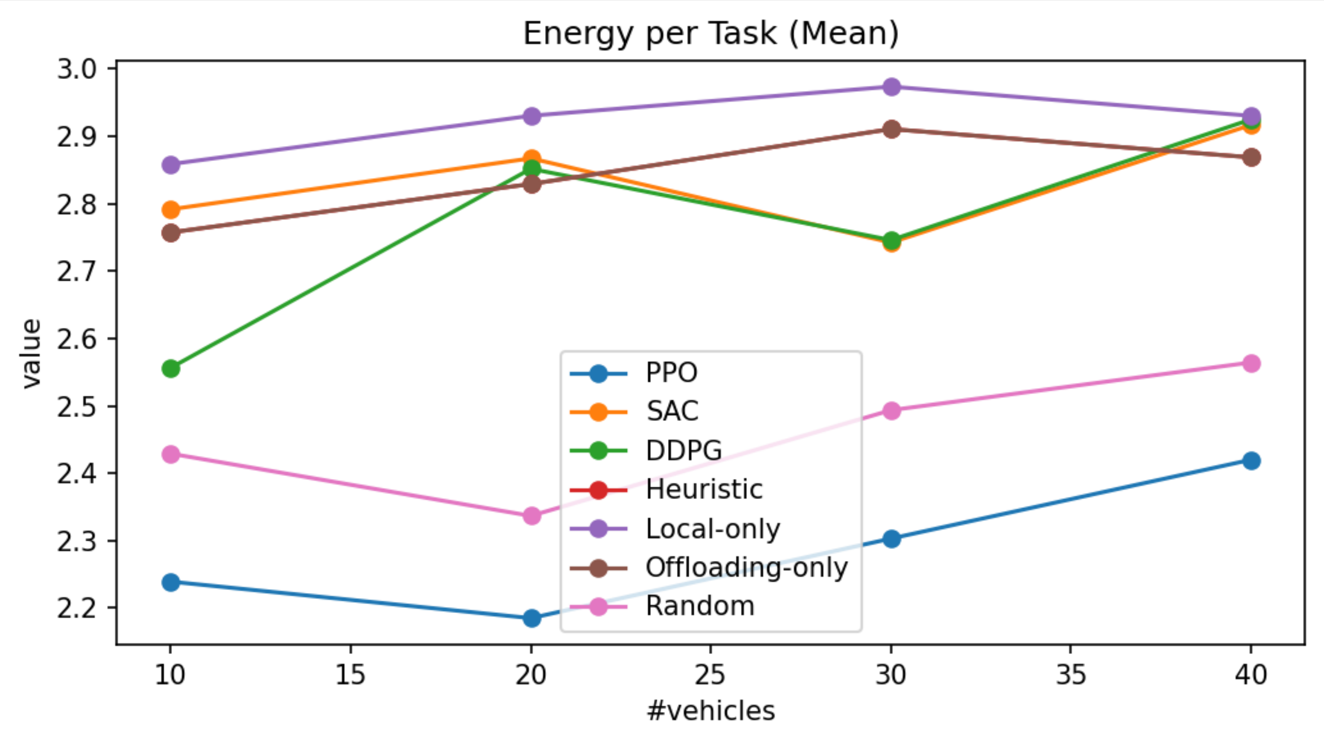
- **Training setup:** horizon 600; n_steps=2048, lr=3e-4; eval on 5 episodes;



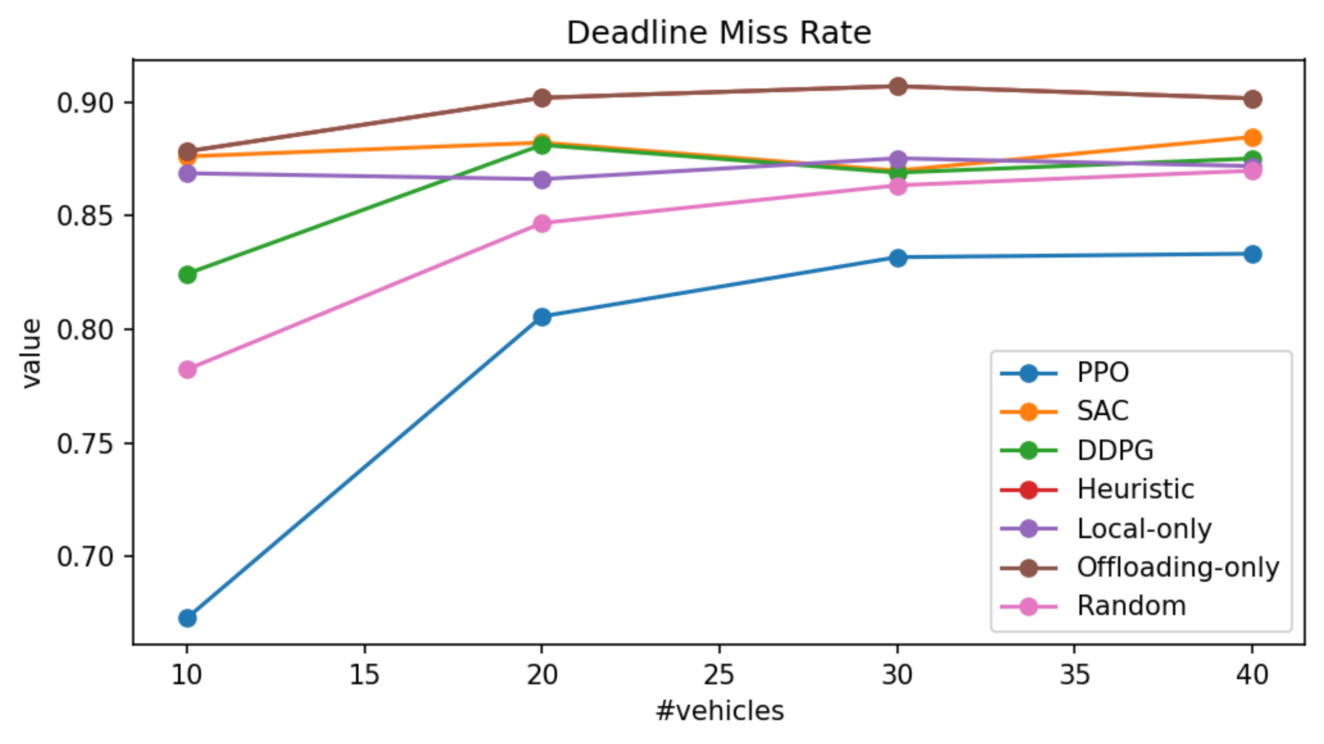
Results

- **Setup:** configs/default.yaml, horizon 600, Poisson arrivals, distance-based power
- **Key findings (qualitative):**
  - **PPO** achieves the **lowest deadline miss rate**
  - Energy per task stays competitive with SAC/DDPG and lower than fixed baselines at load
- **Why:** PPO’s discrete offload + priority decisions adapt to bursty queues and mobility.

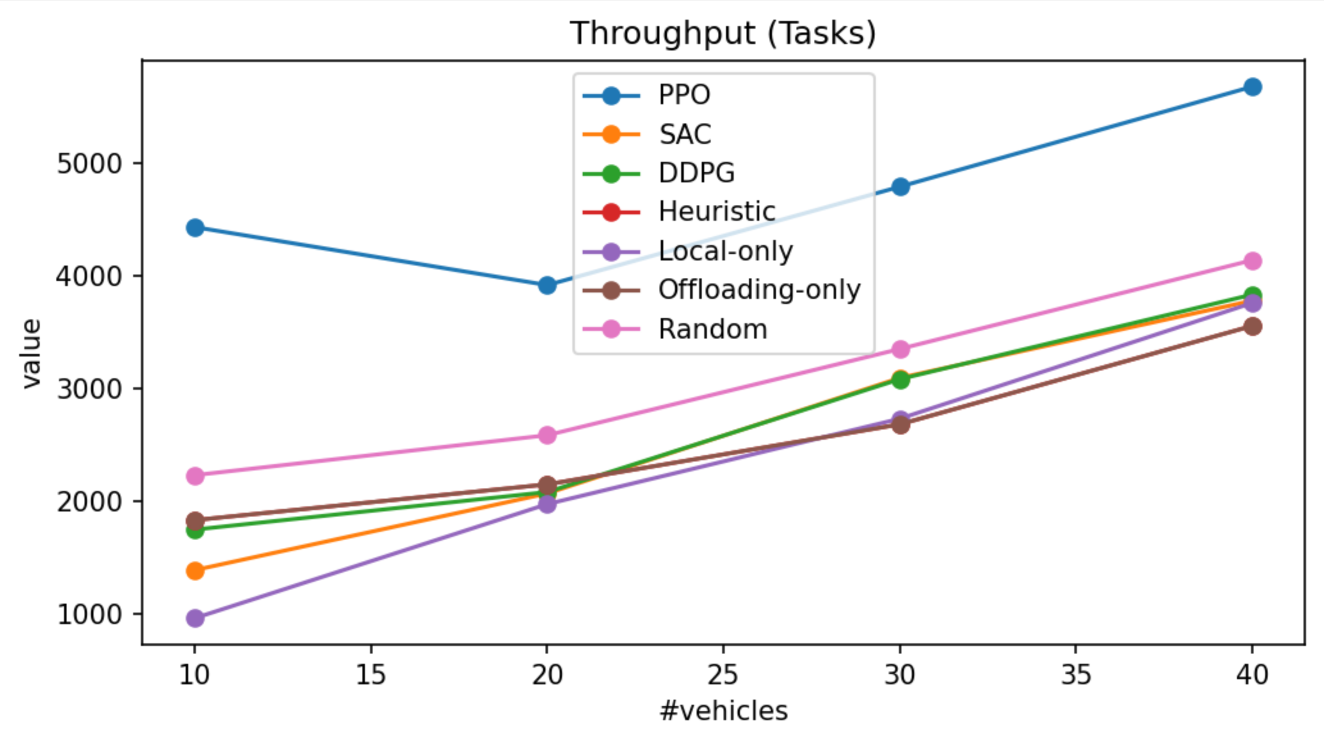
Energy per Task Comparison



Deadline Miss Rate Comparison



Throughput Comparison



Scalability vs #Vehicles

- **Experiment:** $V \in \{10, 20, 30, 40, 50\}$, seeds = $\{0,1,2\}$, 5 eval episodes each.
- **Trend highlights:**
  - As **vehicles↑**, **offloading-only** degrades (shared V2I), **local-only** saturates on compute
  - **PPO** maintains **lower miss rate** by balancing local/V2V/edge/cloud
  - Throughput grows with V
- **Interpretation:** Learning when **not** to offload is as important as offloading

Insights

- **Distance-based power (default) vs learned power (ablation):**
  - Removing the power head shrinks the action space → faster, stabler PPO
- **No-V2V ablation:**
  - Increases queueing at edges/cloud
- **Heuristic (EDF + min-ETA):**
  - Strong at light load; under bursty arrivals and higher V, it misjudges congestion vs PPO
- **Reward normalization (running z-scores):**
  - Critical for stable training on CPU (prevents advantage blow-ups)
- **Takeaway:** The **offload+priority** structure is the main win

Reproducibility & Engineering

- **One-command pipelines**
  - **Train:** scripts/train.py, train_sac.py, train_ddpg.py
  - **Evaluate & metrics:** scripts/eval_metrics.py
  - **Baselines:** scripts/baseline_heuristic.py --mode {heuristic,local,offload,random}
  - **Grids & plots:** scripts/run_grid.py, scripts/plot_metrics.py, scripts/plot_vs_vehicles.py
- **Configs & seeds**
  - YAML configs (Table-III aligned); **config snapshot** saved in each logdir.
  - Multi-seed runs (≥3) aggregated to mean±std tables/plots.
- **Environment**
  - Gymnasium env, Dict observations

Any Questions?

References

1. P. Li, Z. Xiao, X. Wang, K. Huang, Y. Huang and H. Gao, "EPtask: Deep Reinforcement Learning Based Energy-Efficient and Priority-Aware Task Scheduling for Dynamic Vehicular Edge Computing," in IEEE Transactions on Intelligent Vehicles, vol. 9, no. 1, pp. 1830-1846, Jan. 2024, doi: 10.1109/TIV.2023.3321679.