End-to-end numerico: EdgeNet + Safety + Env + PPO (con Critic/Advantage/GAE)

Setup dello slot (Cap. 1 & 2)

UAV: $u_1 : (f1{=}100, f2{=}3, f3{=}10)$, $u_2 : (80, 2, 5)$. **GS:** $g_1 : (c1{=}100, c2{=}10, c3{=}5)$, $g_2 : (80, 20, 2)$.
Link: $(u_1, g_1) : (w1{=}300, w2{=}150)$, $(u_1, g_2) : (500, 90)$, $(u_2, g_1) : (250, 120)$, $(u_2, g_2) : (400, 70)$.
Nel codice: dataclass `UAV,GS,Link,Slot,Scenario`; generator `gen_synth` (Cap. 2).

Pack delle feature (Cap. 7, `pack_features`)

$$u\_feat = \begin{bmatrix} 100 & 3 & 10 \\ 80 & 2 & 5 \end{bmatrix}, \quad g\_feat = \begin{bmatrix} 100 & 10 & 5 \\ 80 & 20 & 2 \end{bmatrix}, \quad e\_feat = \begin{bmatrix} 300 & 150 \\ 500 & 90 \\ 250 & 120 \\ 400 & 70 \end{bmatrix},$$

`edge_index` $= [(0,0), (0,1), (1,0), (1,1)]$ con indici locali (u_i, g_j).

Actor: EdgeNet.forward (Cap. 7)

Per ogni arco si concatena $[f1, f2, f3, c1, c2, c3, w1, w2]$ e si passa all'MLP `edge_mlp`. Esempi di input:

$$\begin{aligned} x_{u_1 \to g_1} &= [100, 3, 10, 100, 10, 5, 300, 150], \\ x_{u_1 \to g_2} &= [100, 3, 10, 80, 20, 2, 500, 90], \\ x_{u_2 \to g_1} &= [80, 2, 5, 100, 10, 5, 250, 120], \\ x_{u_2 \to g_2} &= [80, 2, 5, 80, 20, 2, 400, 70]. \end{aligned}$$

`notx_mlp` calcola il logit di `NO_TX` usando solo $[f1, f2, f3]$.

Logits simulati (solo esemplificativi):

$\ell(u_1 \to g_1){=}2.0,\ \ell(u_1 \to g_2){=}1.0,\ \ell\_\mathrm{NO}(u_1){=}0.2; \quad \ell(u_2 \to g_1){=}1.5,\ \ell(u_2 \to g_2){=}0.5,\ \ell\_\mathrm{NO}(u_2){=}0.1.$

Softmax per-UAV. Per u_1: $\mathrm{softmax}([2.0, 1.0, 0.2]) \approx [0.63, 0.23, 0.14]$; per u_2: $\mathrm{softmax}([1.5, 0.5, 0.1]) \approx [0.54, 0.24, 0.22]$. Output: `probs_dict` e `raw_logits_store`.

Safety layer (Cap. 6, `safety_project`)

Normalizza capacità con `cap_scale`=40:

$$gs\_left[g\_1]{=}\tfrac{100}{40}{=}2.5, \quad gs\_left[g\_2]{=}\tfrac{80}{40}{=}2.0.$$

Fabbisogno link: $need(u \to g) = \frac{w2}{40}$:

$need(u_1 \to g_1) = 3.75\,(>2.5) \ \Rightarrow$ scarta, $\quad need(u_1 \to g_2) = 2.25\,(>2.0) \ \Rightarrow$ scarta,
$need(u_2 \to g_1) = 3.0\,(>2.5) \ \Rightarrow$ scarta, $\quad need(u_2 \to g_2) = 1.75\,(\leq 2.0) \ \Rightarrow\ u_2 \to g_2,\ gs\_left[g_2] \leftarrow 0.25.$

Mappa finale nello slot: $\{u_1 : \mathrm{NO\_TX},\ u_2 : g_2\}$. L'ordine è per probabilità decrescente per UAV, soggetto a vincoli.

Ambiente e Reward (Cap. 4, `DataComEnv.step`)

Metriche: throughput, latency, distance, handover, violations. Servito per UAV: served$(u) = \min(f1, w2)$. Per u_1: NO_TX. Per $u_2 \to g_2$: $f1$=80, $w2$=70 $\Rightarrow$ 70.

$$\text{throughput} = 70, \quad \text{latency} = c2 + \max(0, f1 - \text{served}) \cdot 0.05 = 20 + 10 \cdot 0.05 = 20.5,$$

$$\text{distance} = 400, \ \text{handover} = 0, \ \text{violations} = 0.$$

Pesi: α=1, β=0.02, γ=0.0015, δ=0.3, penalty=50.

$R = \alpha\,\text{throughput} - \beta\,\text{latency} - \gamma\,\text{distance} - \delta\,\text{handover} - \text{penalty}\cdot\text{violations} \approx 70 - 0.41 - 0.6 = 69.0.$

Critic, TD-error, Advantage e GAE (Cap. 7 & 8)

Critic. Stima $V(s)$ da mean$(u_feat)\|$mean(g_feat) via `PPOPolicy.forward_value`. Esempio: $V(s) = 68.5$.

Connessione $A = (Q - V)$, TD-error e GAE. Definizione concettuale:

$$A_t = Q(s_t, a_t) - V(s_t).$$

Approssimazione pratica 1-step:

$$Q(s_t, a_t) \approx r_t + \gamma V(s_{t+1}) \ \Rightarrow \ A_t \approx r_t + \gamma V(s_{t+1}) - V(s_t) \equiv \delta_t.$$

GAE (riduce varianza con parametro $\lambda \in [0, 1]$):

$$A_t^{(\lambda)} = \delta_t + \gamma\lambda\,\delta_{t+1} + \gamma^2\lambda^2\,\delta_{t+2} + \dots$$

Espansione (telescopica): ogni δ_k contiene $-V(s_k)$; sommando, molti termini $+\gamma V(s_{k+1})$ e $-V(s_{k+1})$ si cancellano, lasciando

$$A_t^{(\lambda)} \approx \Big(\text{ritorno pesato dei reward futuri}\Big) - V(s_t).$$

Mini-esempio (slot terminale). Con R=69.0, $V(s)$=68.5, $V(s') = 0$, γ=0.98:

$$A = R + \gamma V(s') - V(s) = 69.0 - 68.5 = +0.5.$$

Se episodio multislots, GAE accumula δ_{t+k} con pesi $\gamma^k\lambda^k$ (vedi loop inverso su `adv` in `ppo_train`).

Update PPO (Cap. 8, `ppo_train`)

L'aggiornamento PPO è il passo che addestra la policy neurale usando i dati dei roll-out.

Definizioni preliminari.

- **Mappa**: l'assegnazione completa nello slot, es. $\{u_1 \to g_1,\ u_2 \to NO\_TX\}$.
- **Stato** s: l'insieme delle feature di UAV, GS e link nello slot.
- **Azione** a: una mappa scelta nello stato s.
- $\pi_\theta(a|s)$: probabilità che la policy neurale con parametri θ scelga a in s.

Probabilità vecchia e nuova. Durante il roll-out le mappe a sono campionate dalla policy vecchia $\pi_{\theta_{\text{old}}}$. All'update, la nuova policy π_θ può dare probabilità diverse. Si definisce quindi:

$$\rho = \frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)}.$$

Interpretazione: $\rho > 1$ indica che π_θ ora preferisce a più di prima; $\rho < 1$ meno di prima.

Obiettivo dell'attore (clipped). L'update ideale sarebbe:

$$\mathcal{L}_\pi = -\mathbb{E}[\rho\, A].$$

Ma ρ può esplodere. PPO introduce il *clipping*:

$$\mathcal{L}_\pi = -\mathbb{E}\big[\min\big(\rho A,\ \text{clip}(\rho, 1-\epsilon, 1+\epsilon)A\big)\big] - \eta\, \mathcal{H}(\pi).$$

- Se $A > 0$: aumenta $\pi_\theta(a|s)$, ma non oltre $1+\epsilon$. - Se $A < 0$: la riduce, ma non sotto $1-\epsilon$. - $\mathcal{H}(\pi)$ è entropia: incentiva esplorazione.

Obiettivo del critic. Il critic stima il ritorno atteso. Il target è:

$$R^{(\lambda)} = A^{(\lambda)} + V(s),$$

ossia ritorno stimato dal GAE più baseline $V(s)$. La loss è:

$$\mathcal{L}_V = \text{MSE}(V_\theta(s), R^{(\lambda)}).$$

Ottimizzazione complessiva. La loss finale combina attore e critic:

$$\mathcal{L} = \mathcal{L}_\pi + 0.5\, \mathcal{L}_V - \eta\, \mathcal{H}.$$

Si fa backpropagation, clipping dei gradienti e ottimizzazione con Adam. Così la policy impara a dare più probabilità alle mappe con $A > 0$ (meglio del previsto) e meno a quelle con $A < 0$ (peggio del previsto), mantenendo stabilità.

Riferimenti puntuali al codice

Cap. 1: I/O `load_scenario`, `save_submission`. Cap. 2: `gen_synth`. Cap. 4: `DataComEnv.step`. Cap. 5: `greedy_policy` (teacher IL). Cap. 6: `safety_project`. Cap. 7: `pack_features`, `EdgeNet` (actor), `PPOPolicy` (actor+critic). Cap. 8: `ppo_train` (IL: CE vs greedy; PPO: rollout, GAE, clipped objective, entropy, MSE value). Cap. 9: `run_inference`.