Il problema affrontato riguarda la **gestione dinamica dei collegamenti UAV–GS (U2G)**, in cui più UAV devono essere associati nel tempo a diverse ground station disponibili.

Il problema è affrontato come puro routing.

La rete è modellata come un **grafo bipartito**, in cui un insieme di nodi rappresenta gli **UAV** e l’altro le **ground station (GS)**; gli **archi** descrivono i collegamenti U2G con i relativi feature fisiche (distanza, bitrate, capacità).  
Su questa struttura, una **Graph Neural Network integrata in un algoritmo di apprendimento per rinforzo (GNN + PPO)** apprende — a ogni slot temporale — quale GS associare a ciascun UAV, bilanciando efficienza, capacità e stabilità del sistema.

Il **dataset di riferimento** proviene da un gruppo di studenti cinesi della **Northwest Polytechnical University (NWPU)**. Si tratta di esperimenti reali con tecnologia **UWB (ultra-wideband)** condotti in un campo di volo di circa 20 × 18 m con 7 UAV e 8 stazioni GS fisse. Sono registrati 30 campioni a 10 Hz (≈ 3 s di volo), per quattro diversi **canali UWB (2–5)**. Tuttavia, i canali non rappresentano bande multipath, ma configurazioni diverse dei moduli UWB (due frequenze ≈ 3.99 GHz e 4.49 GHz × due codifiche), utili a valutare l’errore di misura della distanza per ciascuna configurazione.

Agli scopi di questa challange, il dataset offre quattro repliche indipendenti dello stesso scenario che, nel modello sviluppato viene splittato in un dataset per l’addestramento (da ch2– fino al ch4) e la **validazione LOCO (Leave-One-Channel-Out)** per il canale ch5, cioè escludendo un intero canale (ch5) durante il training e utilizzandolo solo in fase di test per verificare la capacità del modello di generalizzare su condizioni simili.

Dal dataset sono ottenute:

* le **posizioni 3D reali** di UAV e GS,
* le **distanze UWB misurate** (UAV→GS).

Tutti gli altri parametri vengono generati sinteticamente per simulare un contesto di rete coerente:

* *f1 (backlog) ~ N(150, 60) → dati in buffer;*
* *f2 (deadline) ~ U(1, 6) → urgenza trasmissione;*
* *f3 = velocità reale derivata dalle posizioni;*
* *c1 (capacità GS) ~ N(120, 30);*
* *c2 (latenza base) ~ N(20, 8);*
* *c3 (coda GS) ~ N(20, 10);*
* *w1 = distanza reale UAV–GS;*
* *w2 = max(1, 320 − 0.25·w1 + N(0, 3)) → bitrate stimato.*

Solo f3 e w1 sono dati fisici; gli altri valori float sono sintetici ma plausibili, servono a introdurre varianza e relazioni statistiche tra feature. Nella challenge potranno essere sostituiti con misure reali (traffico, SNR, capacità radio) mantenendo la stessa logica di posizione e distanza.

**L’algoritmo** è modellato come una **catena di Markov**, cioè una sequenza di stati che rappresentano nel tempo l’evoluzione del grafo UAV–GS. In ciascuno stato, il modello osserva le feature dei nodi (UAV e GS), dei link (distanza e bitrate stimato) e in base allo stato precedente decide la nuova associazione UAV→GS.

La logica è **ibrida**.

* Le **caratteristiche locali**, cioè la convenienza immediata di un singolo collegamento (distanza, bitrate stimato, capacità residua della GS), vengono gestite con un approccio **greedy e capacity-aware**, che ordina i link per efficienza e scarta quelli che superano la capacità disponibile.
* Le **caratteristiche globali**, invece, che descrivono lo stato complessivo della rete (distribuzione dei carichi, code, congestione), sono elaborate da una rete neurale **MLP edge-centric (EdgeNet)**. Quest’ultima concatena le feature di UAV, GS e link e calcola i **logits** per ogni arco del grafo; un’ulteriore piccola MLP (notx\_mlp) produce il logit per l’azione di non trasmettere. Dopo la **softmax per-UAV**, l’output è una distribuzione di probabilità che esprime la preferenza di associazione verso ciascuna GS o verso NO\_TX.

Durante l’addestramento, questa componente neurale costituisce l’**actor** della policy, che viene ottimizzata insieme a un **critic**. L’actor (EdgeNet) genera la distribuzione di scelte, mentre il critic — una **seconda MLP più compatta (v\_mlp)** — riceve le medie delle feature di UAV e GS e stima il valore atteso dello stato . La differenza tra la reward effettiva e il valore previsto forma l’Advantage, definito come:

che misura quanto l’azione eseguita ha reso meglio () o peggio () rispetto alle aspettative del critic. Questo valore guida la correzione della policy: l’actor aumenta la probabilità delle azioni con , mentre il critic riduce l’errore sulla stima , stabilizzando progressivamente la policy.

L’apprendimento avviene in due fasi complementari, che si susseguono in modo continuo.  
Nella prima, di **Imitation Learning (IL)**, la rete apprende per imitazione da un *teacher greedy*: un selettore deterministico che, per ogni UAV, ordina i link in base all’efficienza e assegna la GS migliore compatibilmente con la capacità residua.  
Questo teacher è *capacity-aware* e integra un meccanismo di **isteresi anti-handover**, che privilegia la continuità del collegamento finché la qualità resta accettabile.  
Lo scopo di questa fase è fornire alla policy una base iniziale stabile e coerente, da cui partire per il successivo apprendimento per rinforzo.

Segue la **Proximal Policy Optimization (PPO)**, che raffina progressivamente la policy attraverso *roll-out sequenziali* — simulazioni temporali degli slot che raccolgono le decisioni e le reward del modello — e *aggiornamenti controllati* basati su **clipping-loss**, per limitare variazioni troppo brusche tra un ciclo e l’altro.  
Un termine di **entropia** mantiene invece un minimo di esplorazione, evitando che la rete si blocchi su soluzioni locali.  
In questa fase, *actor* e *critic* vengono aggiornati insieme: il primo modifica la distribuzione delle scelte, il secondo corregge la stima del valore dello stato , così da convergere verso un comportamento più stabile ed efficiente nel tempo.

La **reward** di ciascuno slot rappresenta il guadagno immediato della policy e viene calcolata come:

da cui il modello stima il **vantaggio** tramite *GAE (Generalized Advantage Estimation)*, che tiene conto della correlazione temporale tra slot successivi.  
Questo valore indica quanto l’azione presa ha reso meglio o peggio rispetto alle aspettative del critic: l’*actor* aumenta la probabilità delle scelte con , mentre il *critic* riduce l’errore sul valore previsto .

**Entropia** = misura di *quanto tieni aperte le opzioni*.  
Formula (solo per completezza): .

* Se una scelta domina (0.99/0.01/0.00) ⇒ **bassa entropia** (sei rigido, quasi non esplori).
* Se più scelte sono possibili (0.6/0.4/0.0) ⇒ **più entropia** (un filo di esplorazione).

 Senza entropia, la policy può “**incollarsi**” presto a una GS e **smettere di provare alternative** (magari migliori quando cambia il traffico).

 In PPO minimizziamo:

Quel “” significa: **premia** (un po’) le policy con entropia più alta ⇒ **mantieni un minimo di esplorazione**.  
è piccolo: non ti rende casuale, **solo** evita che tu ti chiuda troppo presto.

Numeri lampo (per capire l’effetto)

* Caso rigido: → (quasi zero: esplori pochissimo).
* Caso “aperto ma non scemo”: → .  
  Con l’entropia nella loss, il training tende verso qualcosa tipo 0.8/0.2 invece che 0.99/0.01: **continui a scegliere g5**, ma **non chiudi la porta** a g2 se le condizioni cambiano.

Cosa NON è

* Non è rumore a caso: **in inferenza** scegli comunque l’azione più probabile.
* Non sostituisce i vincoli: il **safety layer** resta quello che fa rispettare la capacità.

**Addestramento.**  
Durante il training, il modello è eseguito su GPU (CUDA) e segue la pipeline completa:

[Device] Using cuda

[IL-synth] 50/100 loss=6.7220

[IL-synth] 100/100 loss=4.8589

[PPO-synth] ep 10/50 U=9 G=3 Ravg=15.22 loss=1703.651

[PPO-synth] ep 20/50 U=11 G=4 Ravg=95.16 loss=92006.586

[PPO-synth] ep 30/50 U=12 G=4 Ravg=19.47 loss=2735.740

[PPO-synth] ep 40/50 U=14 G=5 Ravg=132.11 loss=137206.844

[PPO-synth] ep 50/50 U=16 G=6 Ravg=193.23 loss=316953.812

I log mostrano la transizione dall’IL, in cui la rete imita il teacher, alla PPO, dove la policy viene raffinata e la ricompensa media cresce progressivamente, segno di un apprendimento stabile e convergente.

**Inferenza.**  
Durante la validazione sul canale reale (ch5), il modello analizza lo stato corrente del grafo UAV–GS e, in base alle **feature locali** (distanza, capacità residua, bitrate) e **globali** (carichi, code, congestione), determina per ciascun UAV la **ground station ottimale**.  
A ogni slot vengono calcolate la reward istantanea e le metriche principali — throughput, latenza, distanza, handover e violazioni di capacità — con il *safety layer* sempre attivo.  
Il riepilogo finale mostra assenza di violazioni e un basso numero medio di handover, segno che la policy ha appreso un comportamento bilanciato e robusto.

Esempio di output:

{

"t": 14,

"map": {

"u0": "g5",

"u1": "g5",

"u2": "g2",

"u3": "NO\_TX",

"u4": "g1",

"u5": "g6",

"u6": "g4"

}

}

Al tempo , l’algoritmo assegna dinamicamente ciascun UAV alla GS più efficiente o decide di non trasmettere, secondo la **policy ottimale appresa** dall’interazione tra rete neurale, componente greedy e meccanismo di rinforzo.