Universita' La Sapienza

Dipartimento di Informatica

Autonomous Networking

Report Secondo Homework

Author:
Giordano Dionisi
1834919
Mattia Lisi 1709782
Michele Spina 1711821

Supervisor:
Prof. Andrea Coletta
Prof.ssa Gaia Maselli

6 dicembre 2021

SAPIENZA
UNIVERSITÀ DI ROMA

Indice

1 Assunzioni

Si presuppone la lettura/comprensione del primo **report.**

2 Starting Point

Partiamo dal miglior algoritmo precedente: **AISG**[1].

Competitor: MGEO → Lavora come il Geographical Algorithm, ma se non ci sono vicini per un drone, allora si ritornerà al depot → E' energy-consuming, ma ottiene ottimi risultati.

2.1 Approccio Epsilon-Greedy

Tutti gli algoritmi utilizzano l'approccio Epsilon-Greedy, perchè precedentemente ha portato risultati eccellenti → L'**Epsilon-Greedy** con probabilità:

- 1-ϵ sfrutta il Reinforcement-Learning;
- ϵ sfrutta l'MGEO modificato per aggiornare i valori del Reinforcement-Learning[2].

2.1.1 Ritorno dell'OIV

Precedentemente l'OIV[3] aveva pessimi risultati → Si è risperimentato perchè il problema è diverso, vista la nuova azione.

OIV = 10 comporta una successiva migliore decisione sub-ottimale, sperimentalmente: c'è un' esplorazione iniziale maggiore[4] → Il learning per K-Armed Bandit/Q-Learning è migliore.

Ciò non avveniva precedentemente vista l'assenza del ritorno al depot.

[1] AI Simple Geo.

[2] Non si sfrutta l'approccio randomico, perchè precedentemente è stato scadente.

[3] Optimistic Initial Value

[4] Come la funzione *epsilon*, anche la funzione di **Reward** è la stessa del primo Homework → Il valore massimo ottenibile è pari a 2

3 Cosa fare? Migliorare l'Esistente

3.1 Da AISG ad AISG_Updated

L'**AISG** non considera il rientro al depot, quindi lo si è modificato → Quando non ci sono vicini si torna al depot.

Così si ha:

1. **Battery-Consuming:** maggiore → Più frequentemente si torna al depot: ciò è Energy-Consuming;
2. **Score:** abbastanza concorrente ad MGEO → E' un ottimo segnale, visto che si sta sfruttando un semplice *K-Armed Bandit* e non *Q-Learning.*

E' un'ottima partenza per i successivi algoritmi.

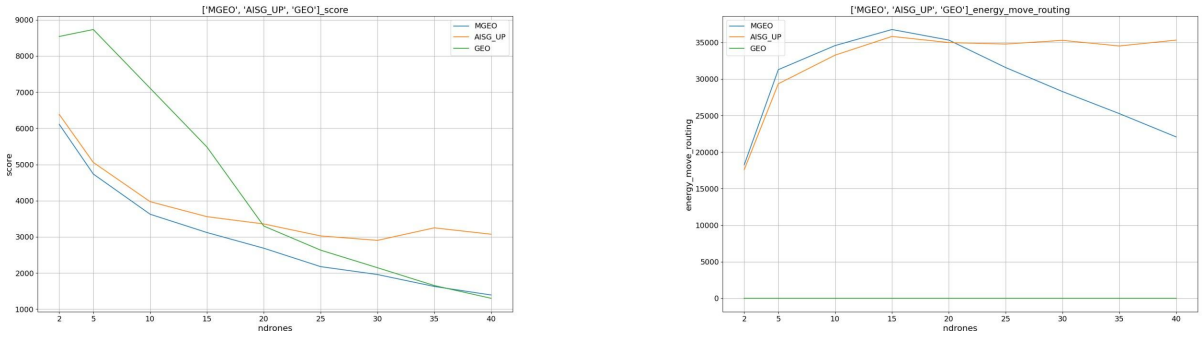

Figura 1: AISG vs AISG_UPDATED vs MGEO vs GEO

Si ha anche il **Geographical Routing** per capire le sue prestazioni → Come AISG, anche il GEO non considera il ritorno al depot: questo lo esclude da molte possibili scelte → Ha un andamento inferiore rispetto all'MGEO ed all'**AISG_Updated.**

3.2 Problemi..?

Anche l'AISG_Updated è lontano dall'MGEO, sopratutto per lo score → Il K-Armed Bandit[5] ha K azioni ed un solo stato per il sistema, quindi non sono conosciuti i precedenti avvenimenti → Ciò crea problemi, perchè si ha un learning molto limitato[6] → Se i droni non tornano neppure al depot, le prestazioni peggiorano.

4 Primi Approcci al Q-Learning

Quindi si passa al **Q-Learning:** è un'estensione del K-Armed Bandit → Sfrutta gli stati e la storia passata e si adatta benissimo, portando risultati sensibilmente migliori.

4.1 Che decisioni prendere? → 2 Actions

Come implementare una base Q-Learning?

Supponiamo di avere μ *droni* → Banalmente si hanno μ *stati.* Per ogni stato[7] si hanno due possibili azioni:

1. Tenere il pacchetto per sè;
2. Passare il pacchetto ad un qualsiasi vicino.

Problema: non sfrutta il ritorno al depot → L'*Energy-Consuming* è bassissimo, ma tantissimi pacchetti scadono, sopratutto con pochi droni.

Ricorda: l'obiettivo principale è lo score piuttosto che il consumo energetivo → Si devono consegnare più pacchetti possibili e perciò 2 Actions non è soddisfacente.

4.2 3 Actions → Torniamo al Depot

Consideriamo una nuova azione: **Tornare al Depot.**

Il consumo energetico aumenta, ma le prestazioni migliorano → Si è ancora distanti rispetto alle prestazioni dell'MGEO

[5] Tipologia di **Reinforcement Learning** utilizzata in AISG Updated

[6] Anche se una **convergenza** più veloce

[7] Ovvero drone, come detto

4.2.1 Esperimento Fallito: N+1 Actions

Tentativo: Usare **N+1 azioni**, ovvero:

- Tenere il pacchetto;
- Andare al Depot;
- Passare il pacchetto ad uno degli N-1 vicini[8]. .

Problema: mancata convergenza → Non riesce ad imparare abbastanza: a fine simulazione ancora non si è arrivati ad avere scelte significative: l'algoritmo sta ancora in esplorazione → Le prestazioni sono disastrose, rispetto agli altri algoritmi.

4.2.2 Esperimento Fallito: Se Conviene Torno al Depot

Tentativo: Se si è entro un range dal depot, si ha almeno un pacchetto e non esistono altre possibilità, allora si va al depot

Idea: Rientrare al depot se non c'è eccessivo consumo → Se si sta dall'altra parte dello scenario, si consuma troppa energia e si evita. In tal caso si:

- Aspetta un drone vicino;
- Mantiene il pacchetto.

Nessun buon risultato → Se un drone è lontano dal depot e non ha vicini, tutti i suoi pacchetti scadono. Essendo lo score l'obiettivo, si ha un tentativo fallito.

5 Svolta: 3A_TMGEO

Definiamo ξ un *drone,* ζ il suo *buffer,* η i *pacchetti* ed α un suo *vicino.* Le seguenti migliorie generano ottimi risultati:

[8] Non obbligatoriamente tutti contemporaneamente visibili in un certo istante.

- Se in ζ vi è almeno un η in scadenza, allora ξ torna al depot, per evitare che η venga perso e che la **Delivery-Ratio** aumenti (come lo **Score**);
- Se ξ ha η ed α sta andando al depot, allora ξ gli passa η, perchè arriverà velocemente, rispetto a cercare altre trasmissioni eventuali.
- Se ξ ha η ed α è nello stato *GoMustBack*, allora ξ gli passa η $\rightarrow$ Con molta probabilità η arriverà velocemente, rispetto a cercare altre trasmissioni eventuali.

  Spiegazione *GoMustBack*: ξ deve ritornare al depot, quindi entra nello stato *GoMustBack*[9].

  Un drone:

  - Va in *GoMustBack*. Per esso:
    - Prosegue lungo la propria traiettoria finchè si sta avvicinando al depot.[10].
    - Contrariamente torna al depot.
  - Effettua la scelta migliore secondo il Q-learning.

  Così si massimizzano gli spostamenti utili del drone, minimizzando la lunghezza dei percorsi da e verso il depot.

Si generano ottimi risultati:

[9] Non è uno stato del sistema, quindi per il Q-Learning, ma uno stato personale del drone

[10] Possibile osservarlo calcolando l'angolo tra traiettoria del drone e traiettoria che dovrebbe percorrere per tornare al depot.

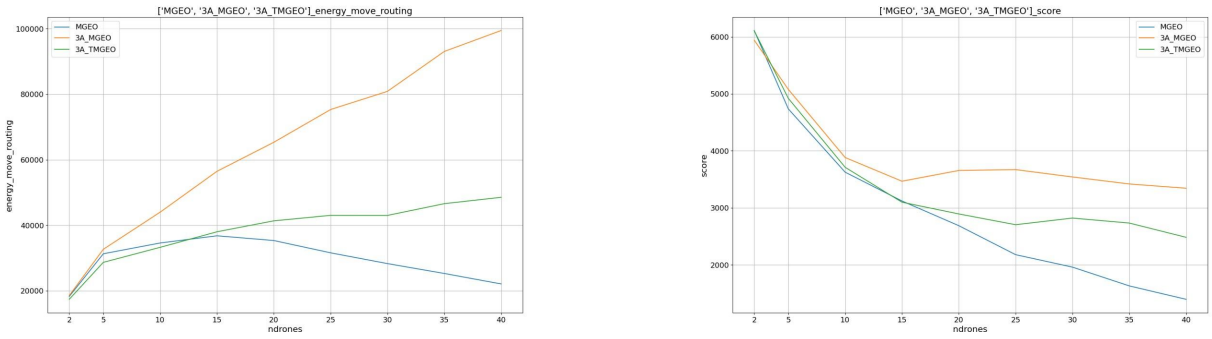

Figura 2: 3A_TMGEO vs 3A_MGEO vs MGEO

5.1 Novità: Grid Algorithm

Considerare i droni come stati comporta una difficile convergenza → Troppi stati.

Non si tiene in considerazione la **posizione** del drone ed il passaggio di stato dipende dal passaggio del pacchetto da un drone all'altro → Ma non è importante il drone come soggetto: è importante la sua posizione → Se due droni si trovano nella stessa posizione, essi sono equivalenti.

Nel Grid Algorithm si divide lo scenario in β celle → Ogni cella è uno stato.

Un drone prende scelte diverse a seconda della sua posizione.

5.1.1 Miglioria: GRID_W_NEXT_UP

Il Grid Algorithm utilizza come chiave per il Q-SET[11] l'indice della cella → **Approccio debole:** non considera la direzione dei droni.

Il **GRID_W_NEXT_UP** sfrutta sia l'informazione della propria cella e sia l'informazione della cella che verrà attraversata dalla sua traiettoria. Ciò porta migliorie, perchè si considera anche la direzione: non si considera il *Next_Target* per evitare una mancata convergenza[12] → La **convergenza** è abbastanza veloce, poiché ogni cella ha dai 2 ai 4 vicini[13]. Al contempo adotta i principi del 3A_TMGEO.

[11] Quindi come stati.

[12] Si è provato anche tale approccio ma è stato fallimentare.

[13] Ciò non genera troppi stati.

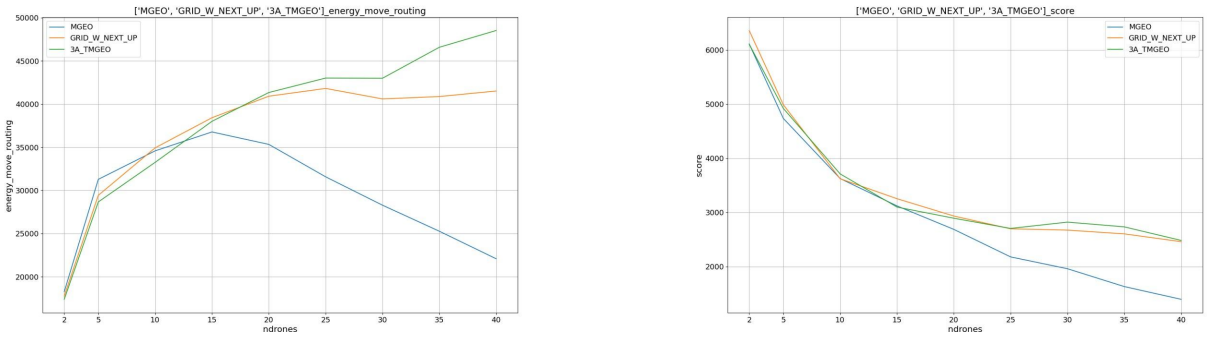

Figura 3: GRID_W_NEXT_UP vs MGEO vs 3A_TMGEO

- **Score:** L'andamento è vicino all'MGEO, con pochi droni → MGEO rientra spesso al depot, mentre GRID_W_NEXT_UP e 3A_TMGEO sfruttano il Q-Learning;

  Considerare come stati le celle e non i droni comporta un miglioramento, perchè due droni nella stessa posizione sono equivalenti → Converge prima e quindi impara più velocemente e più precisamente;

- **Energia:** GRID_W_NEXT_UP ha risultati migliori del 3A_TMGEO, visto il diverso concetto di *stato* e perchè si impara più velocemente considerando meno frequentemente il ritorno al depot.

  E' ancora lineare con il numero dei droni rispetto all'MGEO → Si torna frequentemente al depot anche se ci sono molti vicini, solo perchè il Q-Learning dice che è l'azione ottimale, quando è sub-ottimale.

6 Sviluppi Futuri sullo Score/Batteria

Si è svolta anche un'indagine per ridurre al minimo l'**Energy-Consuming** senza perdita di score, o addirittura migliorandolo.

Si sono ridoti i ritorni convogliando i pacchetti su un unico drone quando due o più droni con almeno un pacchetto si incontrano *rightarrow* Approcci usati:

- Convogliare i pacchetti sul drone più vicino al depot;

- Scegliere il drone a seconda del Q-Learning.

Sono approcci interessanti, che abbassano notevolemente l'Energy-Consuming, ottenendo però uno **Score** simile agli altri algoritmi → Sarebbe interessante approfondire questa ricerca in futuro.

7 Sezioni Implementate per Componenti

- **Giordano:** E' il principale responsabile della costruzione e realizzazione dei seguenti algoritmi:
  - **AISG_Updated;**
  - **3A_MGEO;**
  - **2_Actions;**
  - **N+1 Actions;**
  - **Grid Algorithm.**

  Ha inoltre provveduto in maniera principale alla stesura e realizzazione del **Report.**
- **Mattia:** Ha collaborato nelle attività di **ricerca** sugli **sviluppi futuri.**
- **Michele:** Ha ideato e realizzato i seguenti algoritmi:
  - **GRID_W_NEXT_UP;**
  - **3A_TMGEO.**

  Ha collaborato nella realizzazione degli altri algoritmi e del **Report,** realizzato i plot e svolto le attività di **ricerca** sugli **sviluppi futuri.**

8 Conclusioni/Appendice

Il miglior algoritmo realizzato è: **GRID_W_NEXT_UP.**

Di seguito un riassunto degli gli algoritmi proposti con le varie performance (**Score/Energia**)

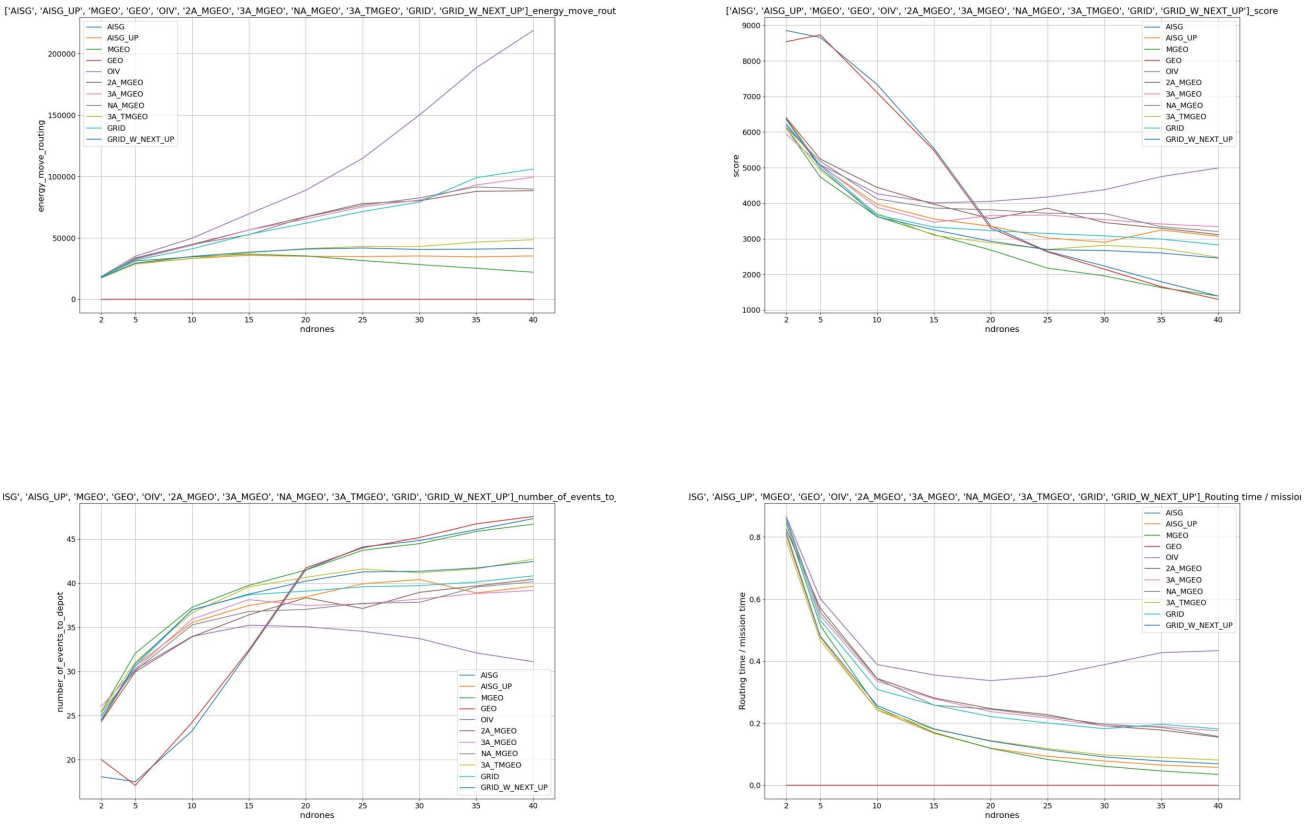

Figura 4: PLOT TOTALE