Modellazione del problema

Dario Bagnara

Contents

1 Modellazione del grafo

In questa prima fase di tesi, lo scopo è **la modellazione**, più generale possibile, del problema. Il posizionamento di un PDC si può ricondurre alla ricerca della posizione ottima di un oggetto generale all'interno di un grafo, considerando alcune metriche specifiche. L'approccio seguito in questa prima settimana è stato quello di cominciare con alcuni **script Python** che definissero in primo luogo un grafo "base", per semplicità con pochi nodi. Per rendere la modellazione abbastanza realistica, ogni nodo è stato dotato delle seguenti proprietà:

- group: gruppo di appartenenza (es cluster)
- level: livello nella gerarchia
- processing: potenza di calcolo
- memory: memoria (in GB)
- storage: spazio di archiviazione (in GB)
- status: stato del nodo (online, offline, maintenance)
- energy: consumo energetico (in kWh)

mentre ogni arco è caratterizzato da:

- latency: latenza della connessione (in ms)
- bandwidth: larghezza di banda della connessione (in Mbps)
- status: stato della connessione (up, down)
- type: tipo di connessione (fiber, ethernet, wireless)

2 Algoritmi

Definito il grafo di partenza, si è poi passati alla definizione, o meglio, alla scelta, di alcuni **algoritmi predefiniti** che identificassero la posizione idealmente ottima in cui posizionare i PDC. Gli algoritmi considerati sono stati:

1. **Approccio greedy (Dijkstra)**: l'algoritmo cerca l'ottimo locale. Partendo da ogni nodo foglia, cerca il cammino più breve per il nodo controllore. Una volta individuato, esamina quest'ultimo controllando le latenze, e, se superiori a una data soglia, piazza un PDC.
2. **Algoritmo randomico**: vengono estratti tutti i nodi tranne quello di controllo, e posizionati in modo randomico n PDC.
3. **Closeness centrality**: l'algoritmo cerca i nodi "più vicini" al nodo di controllo. Fatto ciò, assegna n PDC.
4. **Betweenness centrality**: l'algoritmo cerca i nodi "più centrali", ovvero quelli a metà cammino tra le foglie e il nodo centrale, e assegna a questi n PDC.

3 Visualizzazione

Il grafo a cui sono stati assegnati i PDC è passato a una **funzione di stampa**, che utilizza la libreria matplotlib.pyplot per avere in output un grafo gerarchico. In arancione vengono visualizzati i nodi con PDC, mentre in azzurro quelli sprovvisti.

4 Test

Gli algoritmi vengono messi a confronto tra di loro ma anche realtime con loro stessi, andando a **cambiare parametri** quali latenza e numero di PDC da assegnare. Di seguito, vediamo i risultati ottenuti: L'approccio greedy ha evidenziato come un aumento della latenza su un singolo arco possa influenzare significativamente il posizionamento dei PDC. In particolare, il superamento della soglia massima di latenza lungo un percorso induce l'inserimento di PDC intermedi, con un conseguente aumento del loro numero totale. Questo può introdurre **un overhead** in termini di risorse computazionali e gestione del sistema.

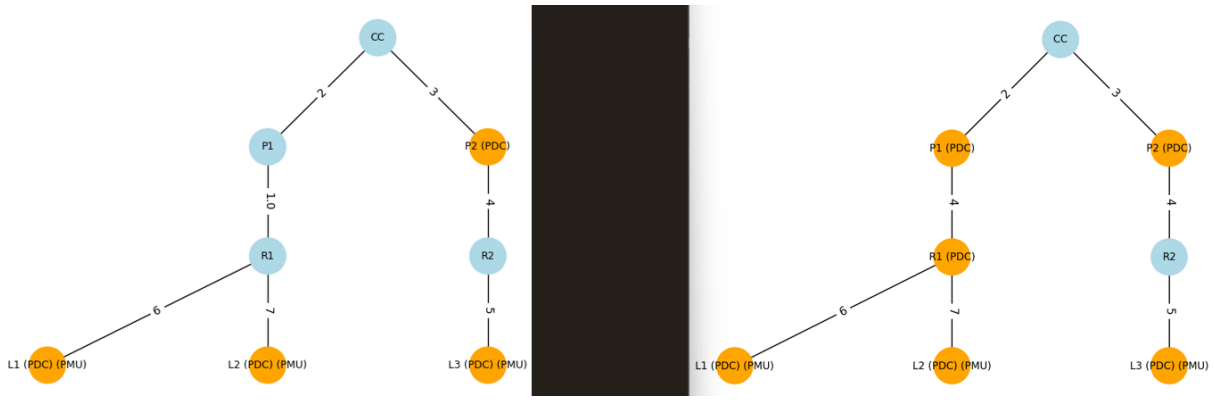

Figure 1: Approccio Greedy

L'approccio randomico, nonostante possiamo dire sia inutile confrontare i risultati ottenuti da un punto di vista esclusivamente visivo, non prevede una strategia accurata per il posizionamento, ma rimane dal mio punto di vista interessante lo studio, nella fase successiva, delle sue capacità di avvicinarsi alla soluzione ottima.

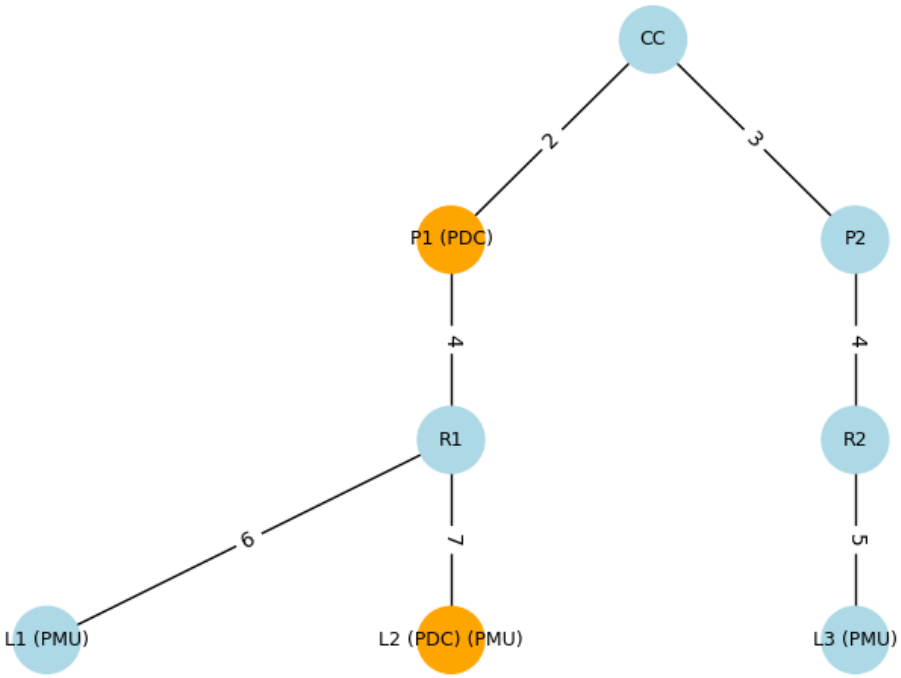

Figure 2: Approccio randomico

L'approccio Closeness centrality ha mostrato che la variazione di latenza apporta una sostanziale modifica, rendendo il grafo molto sbilanciato in quando i nodi più vicini possono essere disposti tutti sul medesimo ramo. Riguardo ciò, si può approfondire la questione **"sbilanciamento" dei PDC** verso nodi collegati con bassa latenza, andando a escludere quelli con latenze elevate in quanto i loro risultati potrebbero essere scartati a causa del funzionamento dei PDC stessi.

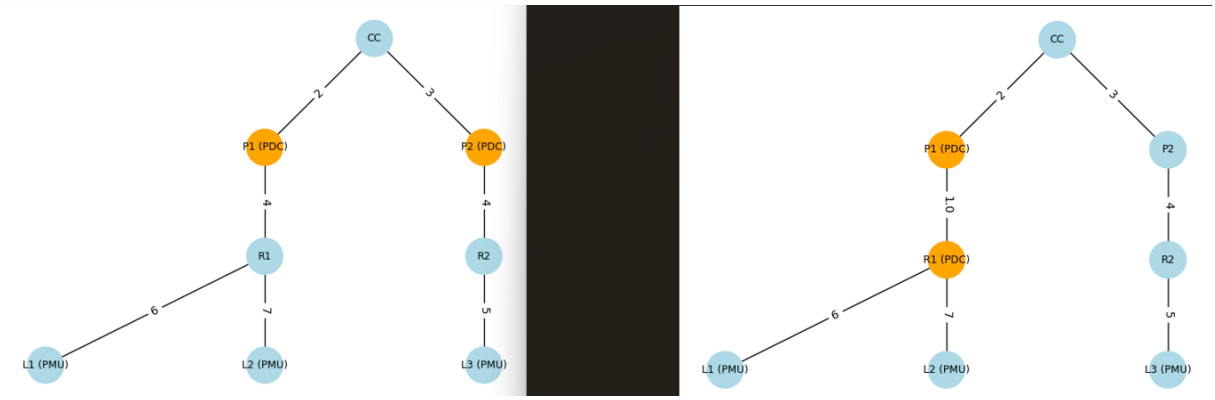

Figure 3: Closeness centrality

L'approccio Betweenness centrality invece non ha mostrato **alcuna differenza** nel cambio di peso. Questo è dovuto prettamente al caso specifico. Va comunque sottolineato che, in questo caso, la centrality viene calcolato usando latency come metrica e non il numero di hop: in un grafo di piccole dimensioni come quello in analisi, i nodi foglia sono comunque "costretti" a passare dai nodi a cui sono stati assegnati i PDC, non avendo altre vie disponibili.

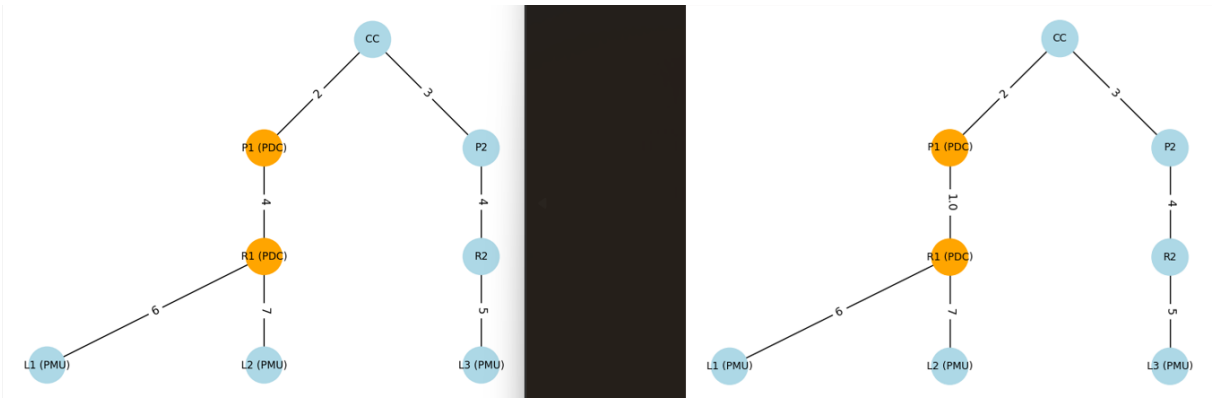

Figure 4: Betweenness centrality

5 Prossimi passi

Il prossimo passo potrebbe essere definire cosa si intende per soluzione ottima/accettabile. Per fare ciò, potrebbe essere utile andare ad individuare gli attributi effettivi di nodi e edge, al fine di avere dei parametri reali (oltre che numerici, anche di "tipologia") su cui poter valutare un algoritmo specifico.