Presentazione Progetto

Introduzione

Il Vehicle Routing Problem si occupa della distribuzione di beni tra depositi e clienti su una rete, distribuzione che viene eseguita tramite un certo numero di mezzi.
Per poter calcolare la soluzione ottima, si deve andare incontro ad un grado di complessità computazionale non indifferente.
Per questo motivo, sono molte le euristiche costruttive (che permettono appunto di costruire una soluzione ammissibile) che sono state realizzate, in modo tale da ottenere soluzioni (che solitamente non sono ottimali) per poter risolvere tali problemi tramite software in maniera efficiente e veloce.
In particolare, il progetto si basa su un CVRP simmetrico, ovvero VRP capacitato, dove per capacità intendiamo la capacità del mezzo di trasporto, mentre l'aggettivo simmetrico si riferisce al fatto che il costo associato all'arco tra due nodi/clienti A e B è uguale a quello tra B e A.

Nella risoluzione del CVRP, vogliamo:
- Creare dei percorsi per ogni veicolo così da servire tutti i clienti
- Non vogliamo superare la capacità di ciascun veicolo
- Vogliamo minimizzare i costi dei percorsi eseguiti

Il problema è composto essenzialmente da clienti, che possono essere suddivisi in due categorie:
- Linehaul: clienti che ricevono materiale
- Backhaul: clienti che forniscono materiale

In più identifichiamo anche un nodo speciale, che è il deposito, da cui partono e ritornano i mezzi di trasporto. I clienti sono visitati da mezzi di trasporto, formando così delle rotte o "route", che appunto indicano la sequenza con il quale i clienti sono visitati.

Il modello del problema che si vuole risolvere è il seguente, modello composto da i dati che identificano il problema, con l'inclusione della funzione obiettivo, da cui vogliamo appunto minimizzare i costi totali di trasferimento, e con la presenza dei vincoli imposti dal problema che devono essere rispettati.
(Vedi modello e vincoli)

Implementazione

Per lo sviluppo del problema, è stato utilizzato Python come linguaggio di programmazione. Possiamo distinguere 3 fasi principali nella realizzazione dell'algoritmo di Clark e Wright:

- Fase preliminare: tramite funzioni di libreria apposite, abbiamo deciso di estrarre dal file le informazioni necessarie per la risoluzione dell'algoritmo, tra cui le informazioni del problema, e i dati sui clienti. In questa fase è stato eseguito il calcolo dei savings, tramite la formula – formula -. I savings permettono di identificare quanto è conveniente spostarsi da un cliente i ad un cliente j. Per questo motivo, abbiamo ordinato in maniera decrescente i saving, così da ottimizzare immediatamente la creazione delle route

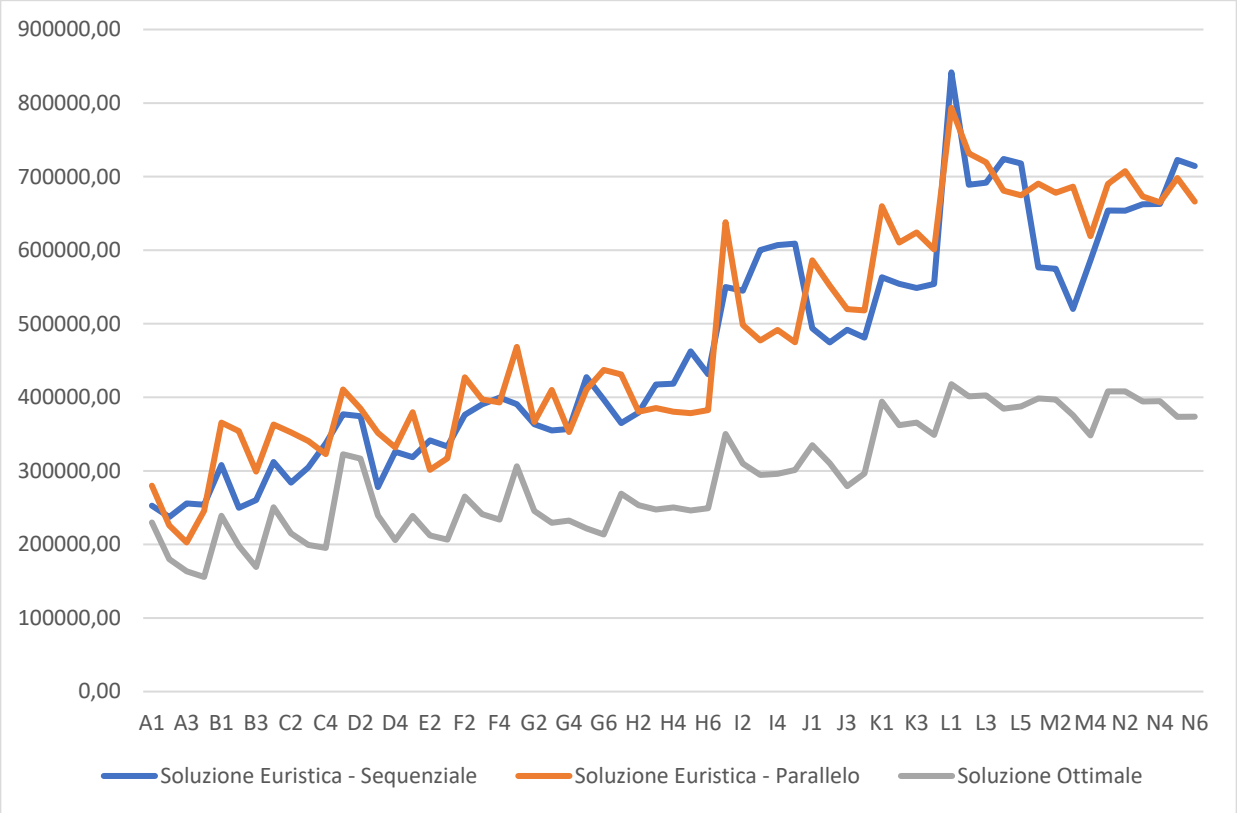
- Applicazione algoritmo: la versione dell'algoritmo di Clarke e Wright è stata proposta in due versioni, quella parallela e quella sequenziale, descritte come segue:
    - Versione Parallela: nella versione parallela dell'algoritmo, abbiamo la possibilità di creare più routes alla volta. Questo vuol dire che l'algoritmo scorre i saving, composti da coppie di clienti (i-j) e verifica la presenza di routes. Se non ce ne sono, viene aggiunta la prima route (0-i-j-0), altrimenti si va a scorrere la lista di routes già inserite e si controlla se il saving che stiamo andando a controllare possa essere effettivamente aggiunto alla route in esame, sia per quanto riguarda la presenza di uno dei nodi del saving, sia per la possibilità di aggiunta in termini di capacità del veicolo. Se una route non ammette quel saving, si va avanti con la prossima route;

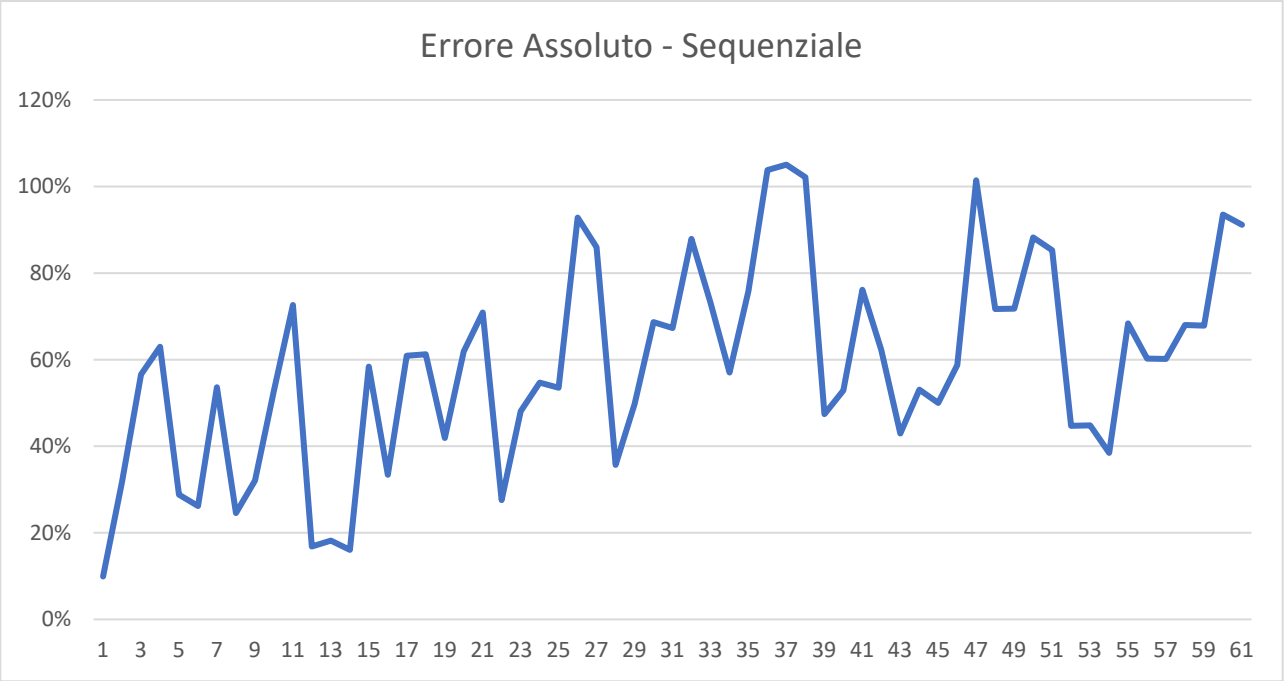
se nessuna route ammette quel saving, esso viene usato per generare una nuova route.

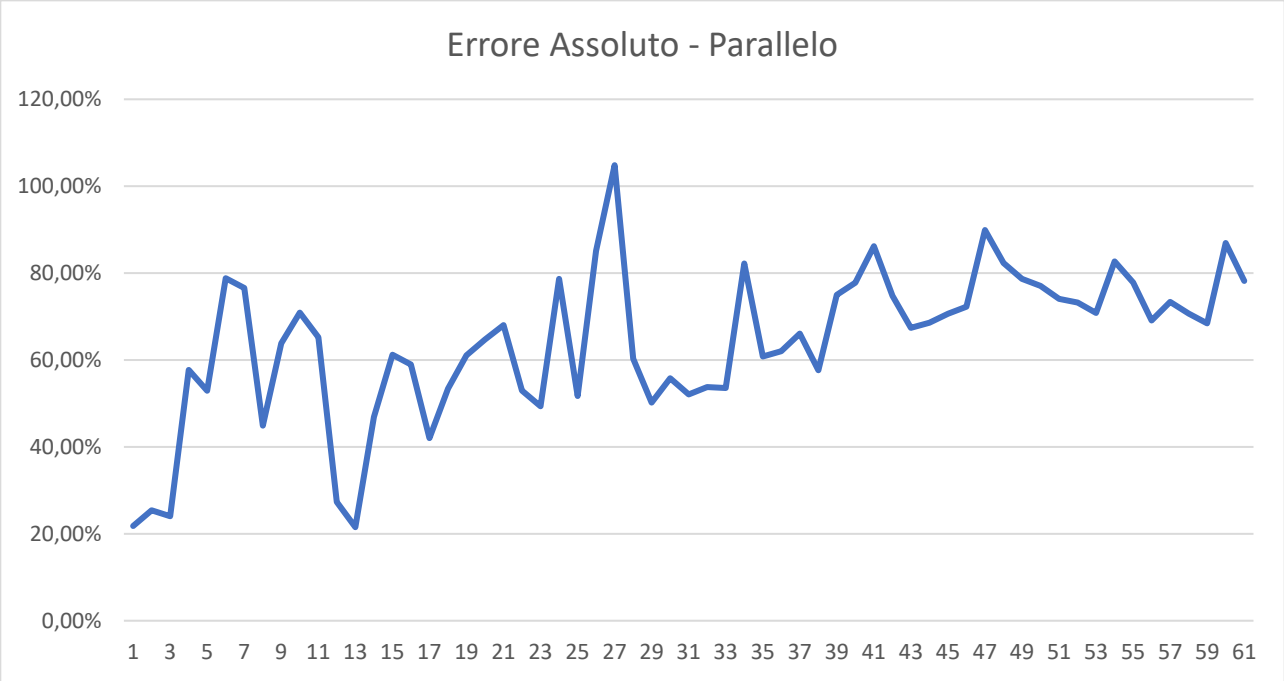
  - Versione Sequenziale: in questa versione, è necessario costruire un percorso alla volta. Il procedimento adottato è il seguente. Per ogni saving, composto da una coppia di clienti (i-j), se ancora non è stata costruita nessuna route, andiamo a costruire la prima route [0-i-j-0], in seguito, si andrà a vedere se la lista di saving contiene clienti che possono essere inseriti in testa o in coda, in maniera tale da servire quanti più clienti è possibile nella route, naturalmente controllando che la capacità del vincolo non venga superata. Quando non è più possibile aggiungere clienti alla route, allora questa è completata e si passa alla creazione della route successiva, servendo i clienti rimasti.

- Merge finale: abbiamo deciso di applicare l'algoritmo separatamente sulle due tipologie di clienti in possesso; prima andiamo a creare le route di soli LH, e poi quelle di soli BH. Questa scelta è stata presa anche per via del vincolo imposto secondo cui le route devono essere composte prima da clienti LH e poi da clienti BH. In questo modo tramite il merge otteniamo le route finali. (Se chiedono, il merge è stato eseguito scegliendo di inserire gli elementi di BH nelle route LH secondo un criterio basato sul miglior saving tra il BH e le possibili code dei LH)

Risultati







Per quanto riguarda i costi, si può notare dal grafico come i costi degli algoritmi in sequenziale e in parallelo siano piuttosto simili, e che viaggiano sulla stessa lunghezza, molto più elevati rispetto ai costi delle soluzioni ottimali. Inoltre, si noti anche come per i primi file, i costi tra sequenziale-parallelo e ottimale siano simili, o per lo meno leggermente superiori, mentre andando avanti con file più pesanti notiamo come la differenza tende ad aumentare. Possiamo dunque affermare che inizialmente i due algoritmi tendono ad ottenere dei risultati discreti, ma con l'aumentare della complessità i risultati che otteniamo sono abbastanza distanti dall'obiettivo ottimale segnato in grigio.

Notiamo inoltre come anche l'errore assoluto sia per sequenziale che per il parallelo sia simile (sequenziale 59%, parallelo 64%), dove in entrambe le soluzioni purtroppo si raggiungono picchi con errori che superano il 100% per alcuni file, mentre per i primi, anche qui l'errore è leggermente minore.

Conclusioni

Per concludere, si può affermare che la nostra progettazione dell'algoritmo Clarke & Wright ha avuto un discreto successo; i vincoli del problema del CVRP sono stati rispettati in pieno e quindi tutti i clienti sono stati serviti senza eccedere la capacità massima dei veicoli.
Abbiamo notato come i costi siano piuttosto differenti rispetto a quelli ottenuti tramite gli algoritmi specializzati, soprattutto per quanto riguarda l'algoritmo parallelo che dovrebbe essere quello che crea routes maggiormente simili a quelle delle soluzioni ottime. Questo significa che l'algoritmo non è completo, manca qualche dettaglio o qualche parte che permetta l'effettivo ottenimento di costi minori; in futuri sviluppi si potrebbe migliorare tramite aggiunta di controlli più efficienti, sia per la versione sequenziale che per la parallela.