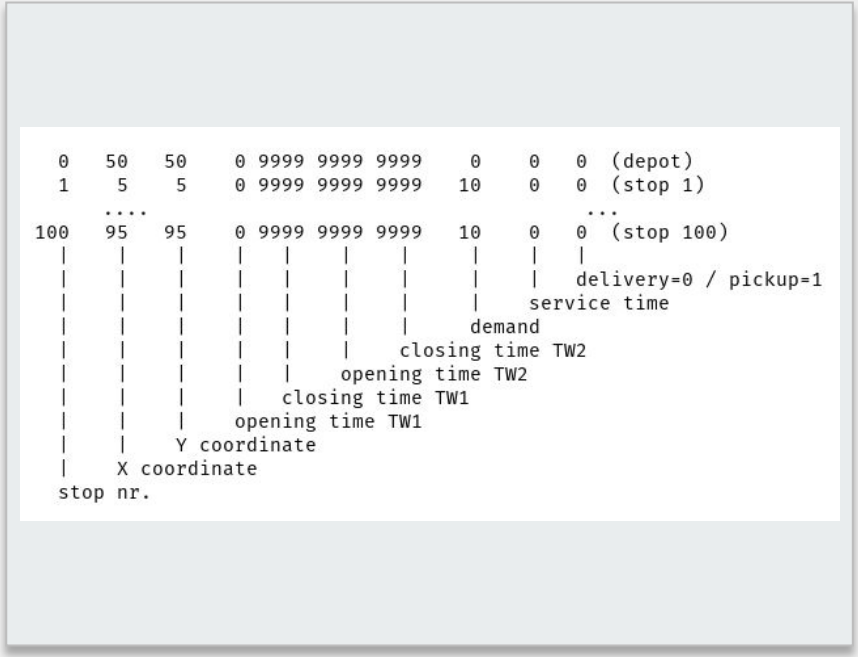
lgvRouting

Luca Bartoli, Massimiliano Bosi
Modena, 24/09/2021

Dataset

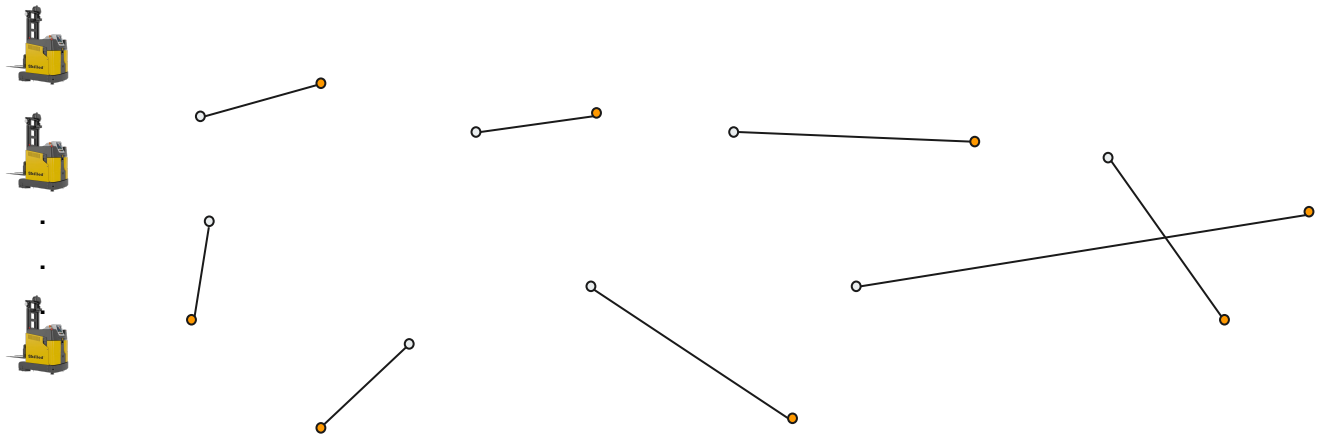
Si è scelto di utilizzare il dataset di VRPPD P2 proposto da Breedam. Il dataset contiene 60 istanze, ognuna di esse formata da 100 stop.

Non sono state valutate la capacità dei singoli lgv e le richieste degli stop poiché si ipotizza di spostare un pallet alla volta.



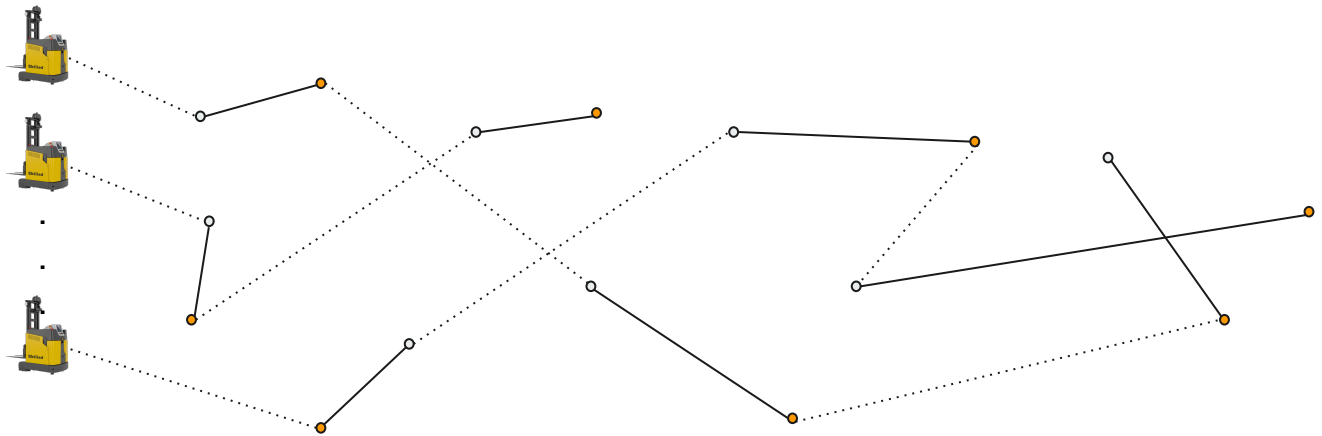
Problem

Date delle coppie di missioni ($start_i$, end_i) ed N navette, dobbiamo trovare il routing ottimale per eseguirle tutte nella distanza minore (di conseguenza in tempo minore).



Problem

Date delle coppie di missioni ($start_i$, end_i) ed N navette, dobbiamo trovare il routing ottimale per eseguirle tutte nella distanza minore (di conseguenza in tempo minore).



Premessa

- ➔ Distanza calcolata secondo euclide. (possibilità manhattan o maximum)
- ➔ Tempi sono calcolati in media sul dataset P2 utilizzando tutti i 60 problemi.
- ➔ Come euristica costruttiva le missioni sono ordinate in ordine crescente di peso.

Algoritmi sviluppati

1. Constructive
2. LocalSearch
3. DepthLocalSearch
4. Multistart (Singlecore - Multicore - GPU)
5. SimulatedAnnealing
6. TabuSearch

Constructive algorithm

- L'algoritmo è sviluppato secondo un euristica costruttiva.
  - Date N navette e M missioni da compiere, l'algoritmo ordina le missioni in base al peso e l'assegna alla navetta con il minor carico.

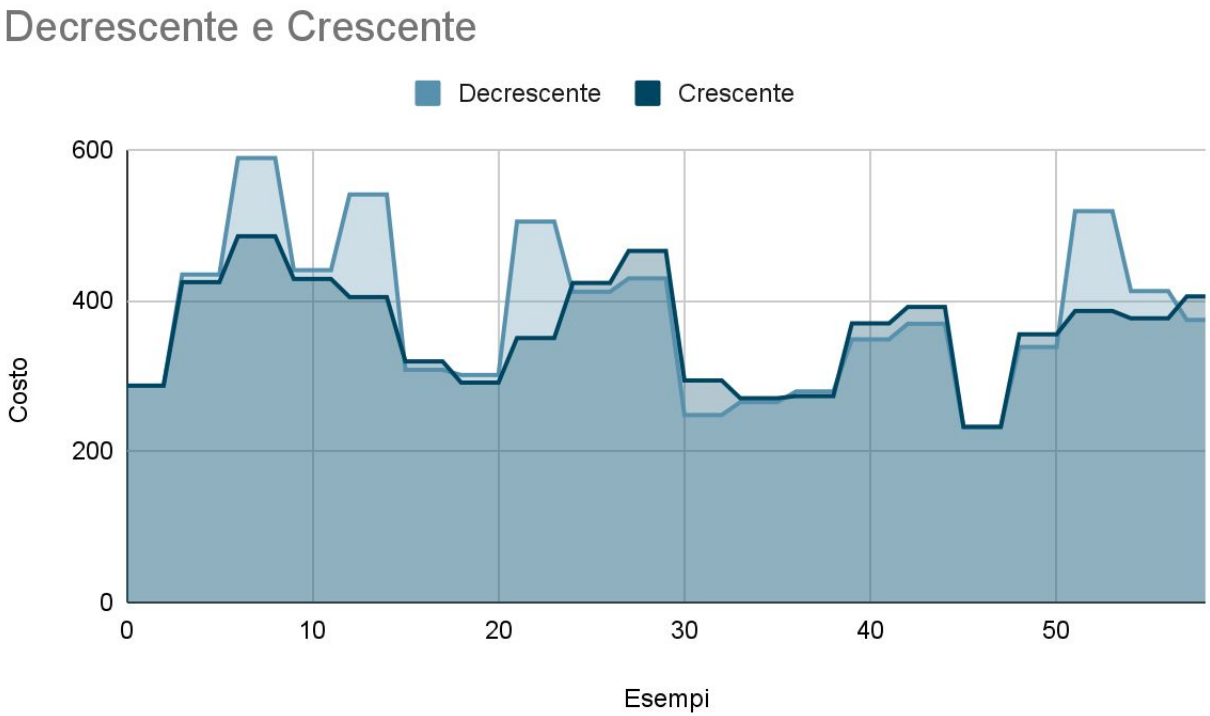
- Parametri:
  - Ordine crescente o decrescente.

Result - Constructive algorithm

L'ordinamento crescente in media funziona meglio.

Avg:

Decrescente: 375.716
Crescente: 353.404



LocalSearch

- Partendo dall euristica costruttiva precedente, l'algoritmo effettua N swap cercando di generare una soluzione migliore fino a che non scatta il timeout oppure si trova una soluzione peggiore.

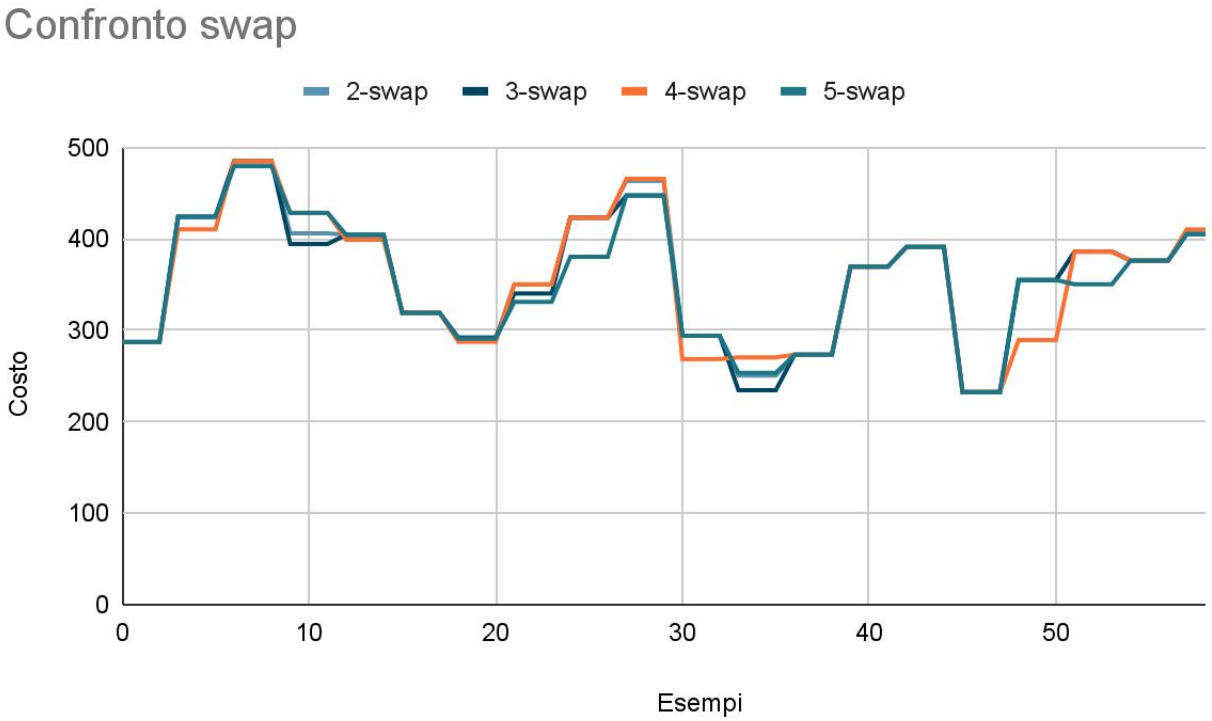
- Parametri:
  - Numero degli swap da applicare alla soluzione iniziale;
  - Timeout.

Result - LocalSearch

Confronto in base al numero di swap applicati. Il timeout non viene considerato in questo test.

Avg:

2-swap: 351.021 - 15ms
3-swap: 349.006 - 16ms
4-swap: 347.652 - 20ms
5-swap: 346.288 - 22ms



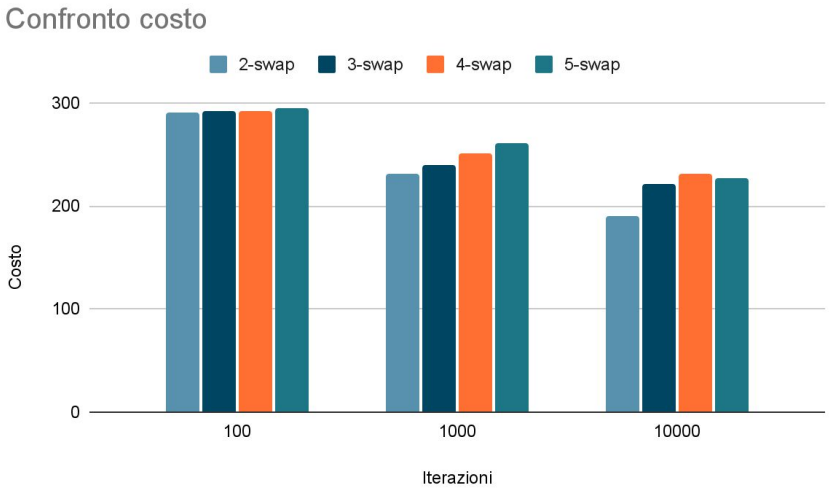
DepthLocalSearch algorithm

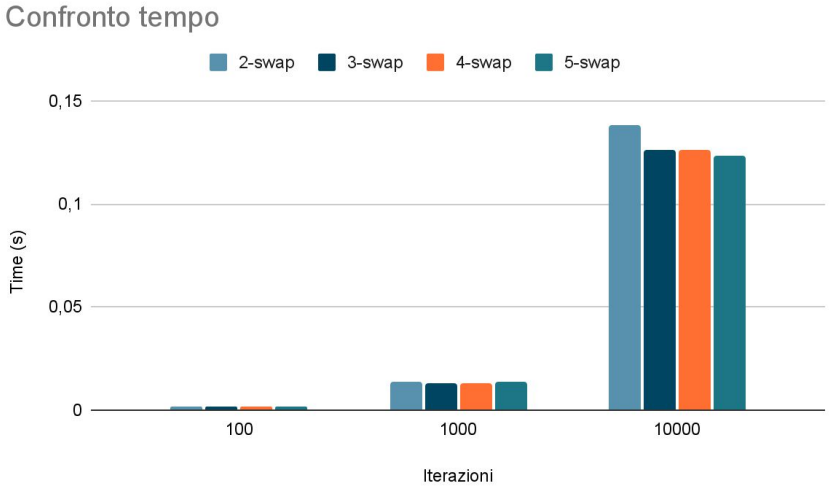
- Modifica alla condizione di terminazione del LocalSearch. Partendo da una soluzione iniziale, si effettuano N swap. Se la soluzione è migliore della precedente si utilizza quella per la prossima iterazione. La condizione di terminazione è data dal numero di iterazioni massime o dal timeout.

- Parametri:
  - Numero degli swap da applicare alla soluzione iniziale;
  - Timeout;
  - Iterazioni massime.

Result - DepthLocalSearch

Confronto sulla base del numero di swap e delle iterazioni





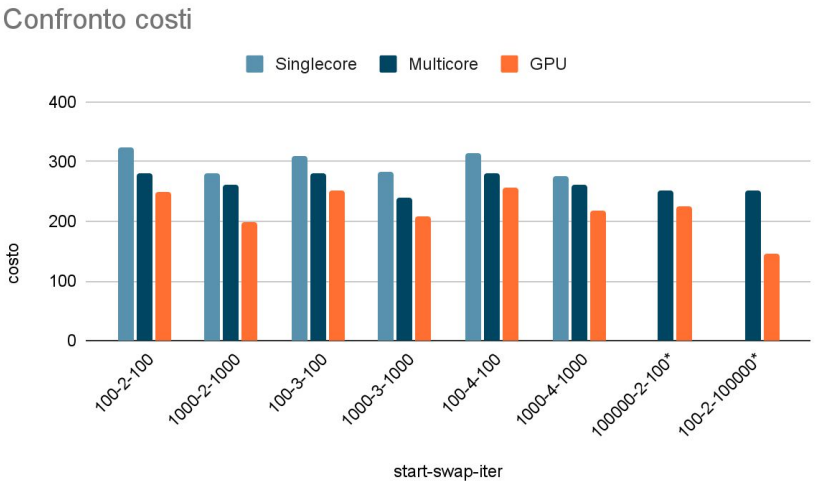
Multistart algorithm

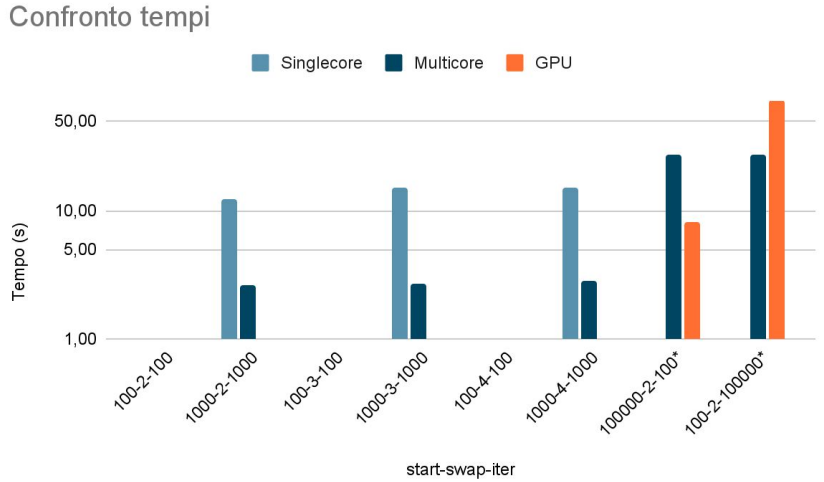
- L'algoritmo genera N_{start} soluzioni casuali, e da queste effettua N_{sw} swap per N_{it} iterazioni.

- Parametri:
  - Numero degli swap da applicare alla soluzione iniziale casuale;
  - Timeout;
  - Numero soluzioni iniziali;
  - Iterazioni da eseguire su ogni soluzione iniziale;
  - SingleCore, Multicore o GPU.

Result - Multistart

Confronto sulla base del numero di start, swap e iterazioni





*I valori su singlecore non sono riportati per il tempo necessario al computamento

SimulatedAnnealing algorithm

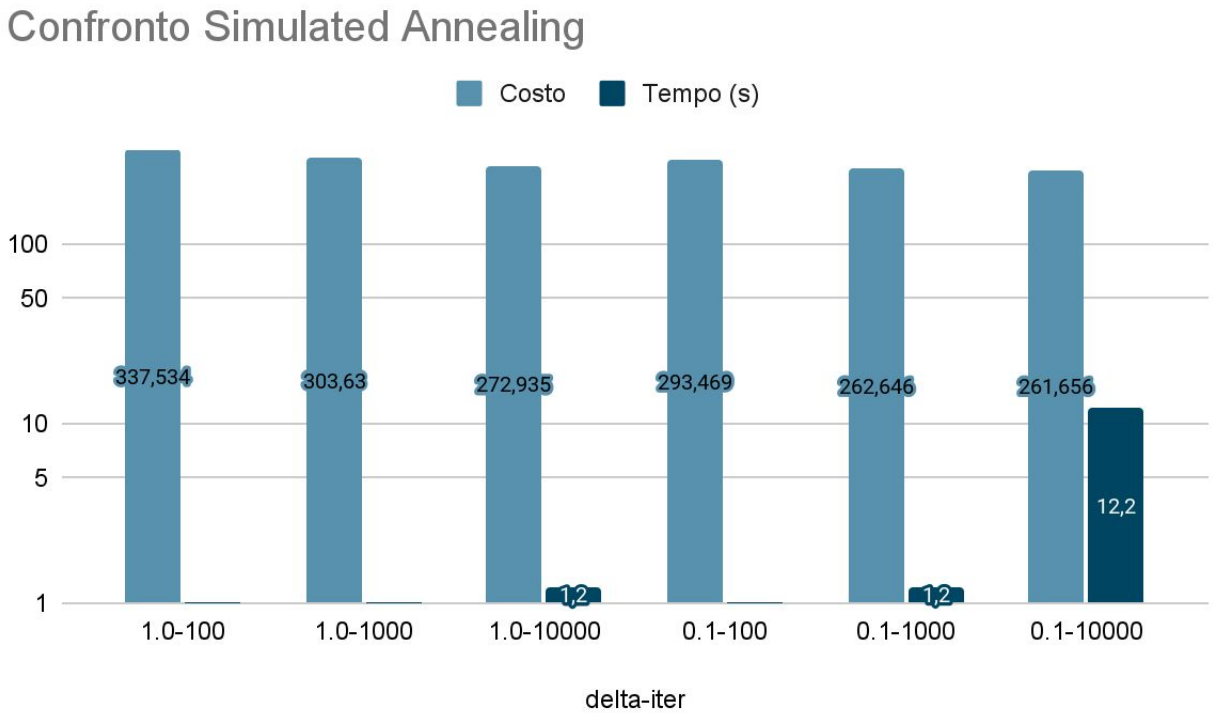
- Partendo da una soluzione casuale ammissibile, l'algoritmo effettua uno swap con una probabilità P(sol) se la soluzione trovata è peggiorativa, altrimenti soluzione migliori sono sempre accettate.

$$P(sol) = e^{\frac{diff(sol, sol_{prec})}{temperature}}$$

- Parametri:
  - Temperatura iniziale;
  - Temperatura minima;
  - Delta di decremento della temperatura;
  - Iterazioni di discesa della temperatura;
  - Timeout.

Result - SimulatedAnnealing

Test eseguiti fissando una temperatura iniziale di 10 e minima di 0.1



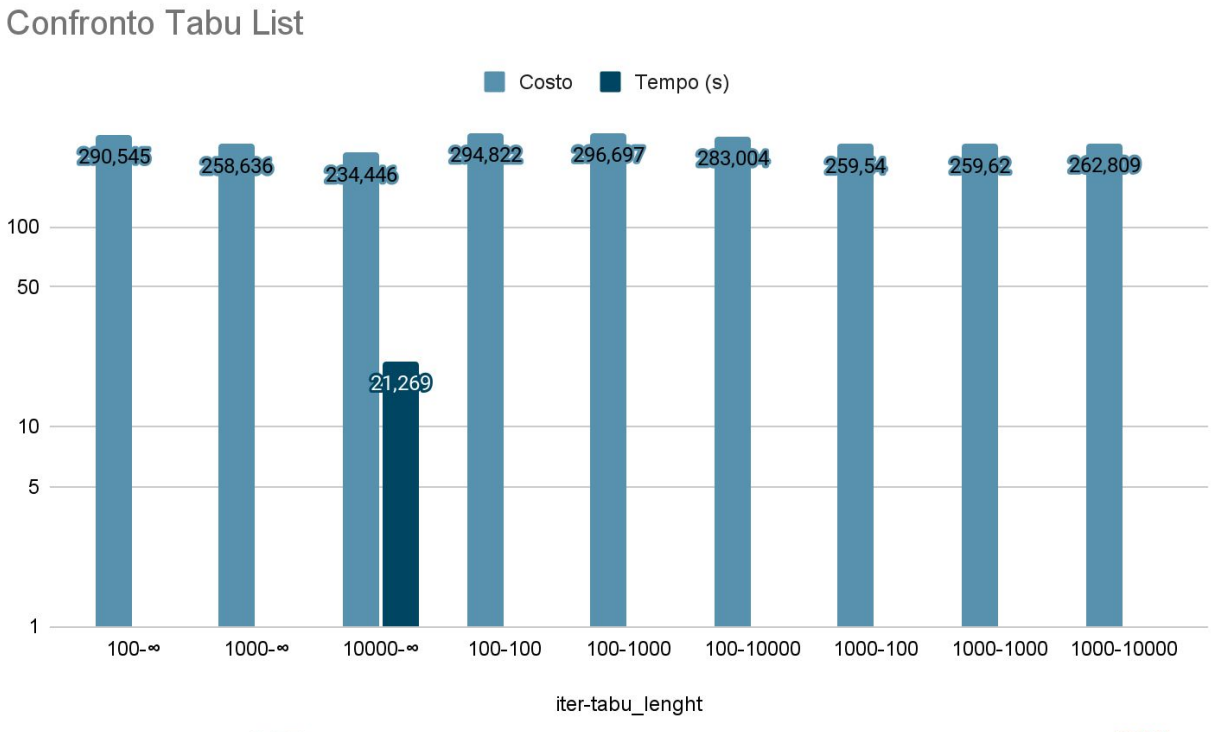
TabuSearch algorithm

- Partendo da una soluzione iniziale tramite euristica costruttiva, l'algoritmo effettua N_{swap} per N_{it} iterazioni prendendo anche soluzioni che si discostano di un certo threshold per scappare da minimi locali. Soluzioni già esplorate vengono evitate tramite l'inserimento in una tabu list con lunghezza configurabile.

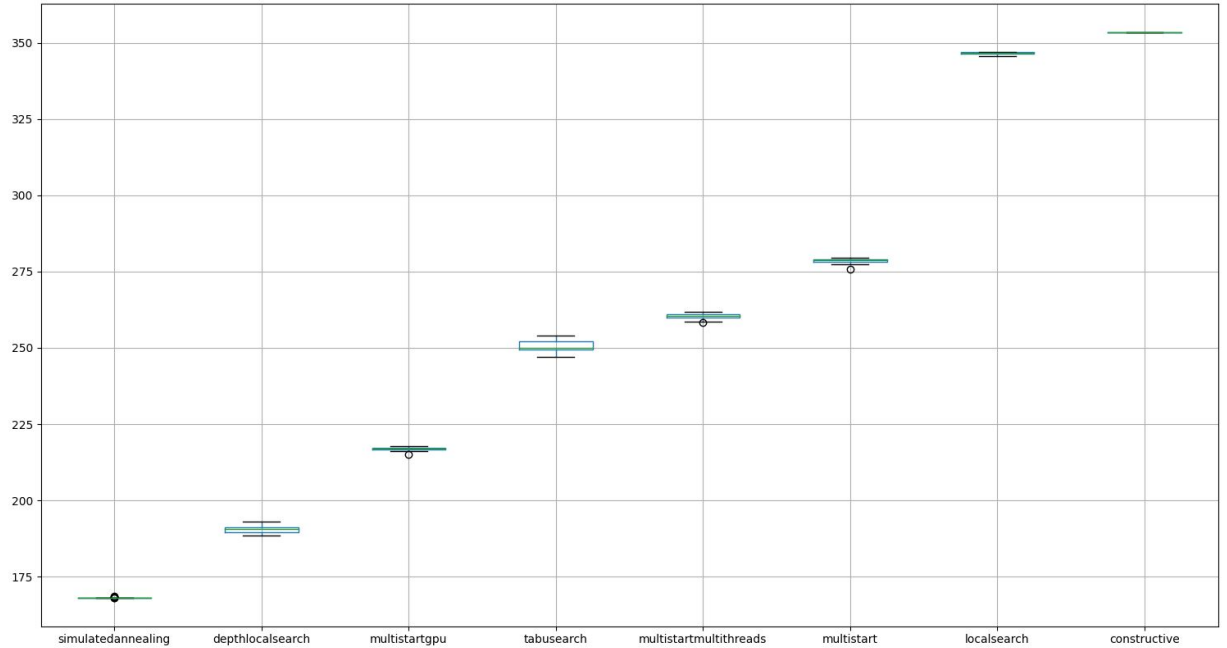
- Parametri
  - Numero degli swap da applicare alla soluzione costruttiva iniziale;
  - Iterazioni dell'algoritmo;
  - Massima differenza di costo per accettare una nuova soluzione;
  - Lunghezza della lista;
  - Timeout.

Result - TabuSearch

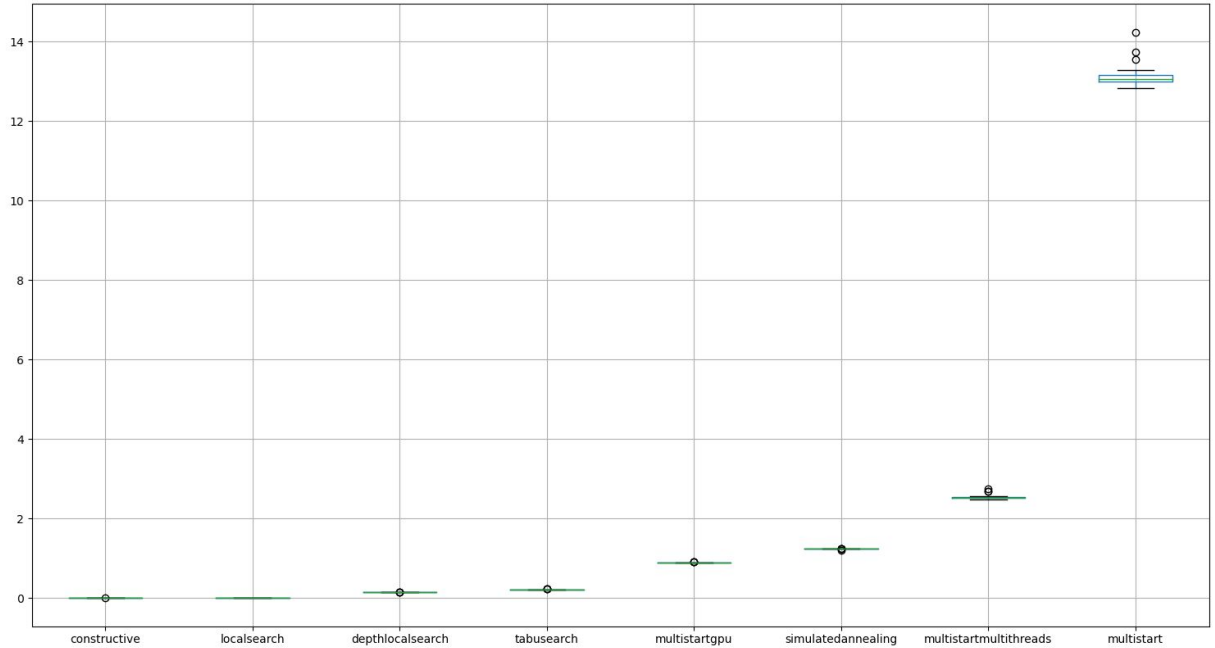
Settato il numero di swap a 3 e la differenza di accettazione a 4



Confronto Costo



Confronto Tempi



Classifica Valutazione

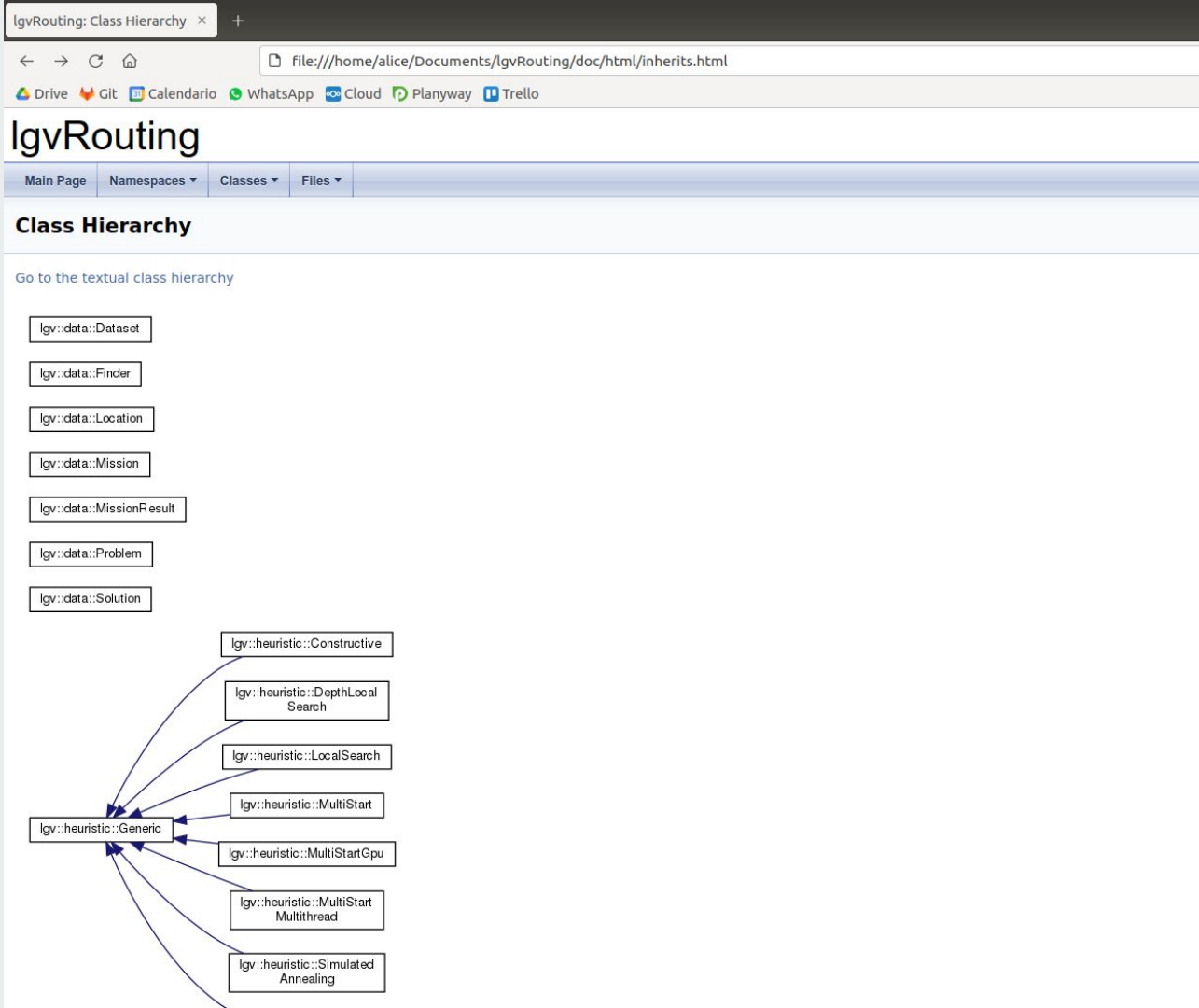
Si stila una classifica in base a AVG_{time} / AVG_{cost}

1. constructive
2. localsearch
3. depthlocalsearch
4. tabusearch
5. multistartgpu
6. simulatedannealing
7. multistartmultithreads
8. multistart

Codice

- C++, CUDA
- Documentazione con Doxygen
- Test con catch2

https://github.com/lucabart97/lgvRouting



Grazie dell’attenzione

Luca Bartoli
228618@studenti.unimore.it
Massimiliano Bosi
205839@studenti.unimore.it