



lgvRouting

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

Dataset

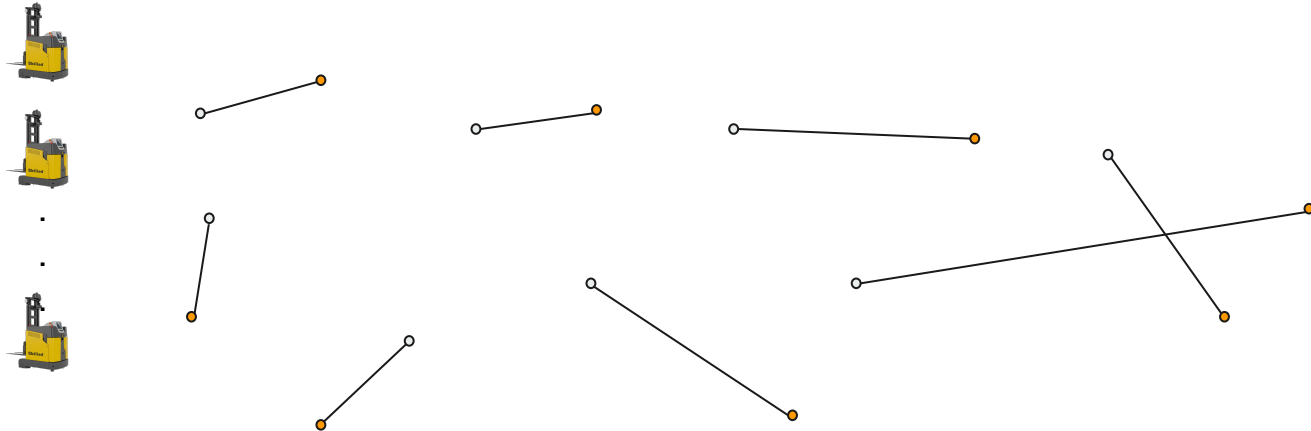
Usiamo il dataset di VRPPD P2 proposto da Breedam. Il dataset contiene 60 istanze, ognuna di esse formata da 100 stop.

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

```
0  50  50  0 9999 9999 9999  0  0  0 (depot)
1   5   5  0 9999 9999 9999 10  0  0 (stop 1)
...
100 95  95 0 9999 9999 9999 10  0  0 (stop 100)
|   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   | delivery=0 / pickup=1
|   |   |   |   |   |   |   |   | service time
|   |   |   |   |   |   |   |   | demand
|   |   |   |   |   |   |   |   | closing time TW2
|   |   |   |   |   |   |   |   | opening time TW2
|   |   |   |   |   |   |   |   | closing time TW1
|   |   |   |   |   |   |   |   | opening time TW1
|   |   |   |   |   |   |   |   | Y coordinate
|   |   |   |   |   |   |   |   | X coordinate
|   |   |   |   |   |   |   |   | stop nr.
```

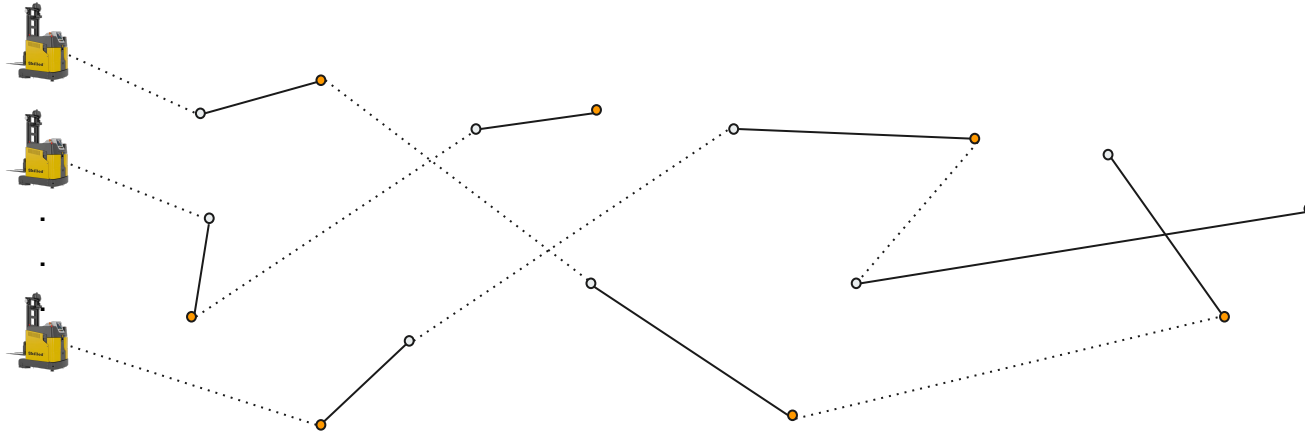
Problem

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).





Algoritmi sviluppati

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



Premessa

- Distanza calcolata secondo euclide. (possibilità manhattan o maximum)
- Tempi sono calcolati in media sul dataset D2 utilizzando tutti i 50 problemi.
- Come euristica costruttiva le missioni sono ordinate in ordine crescente di peso.



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

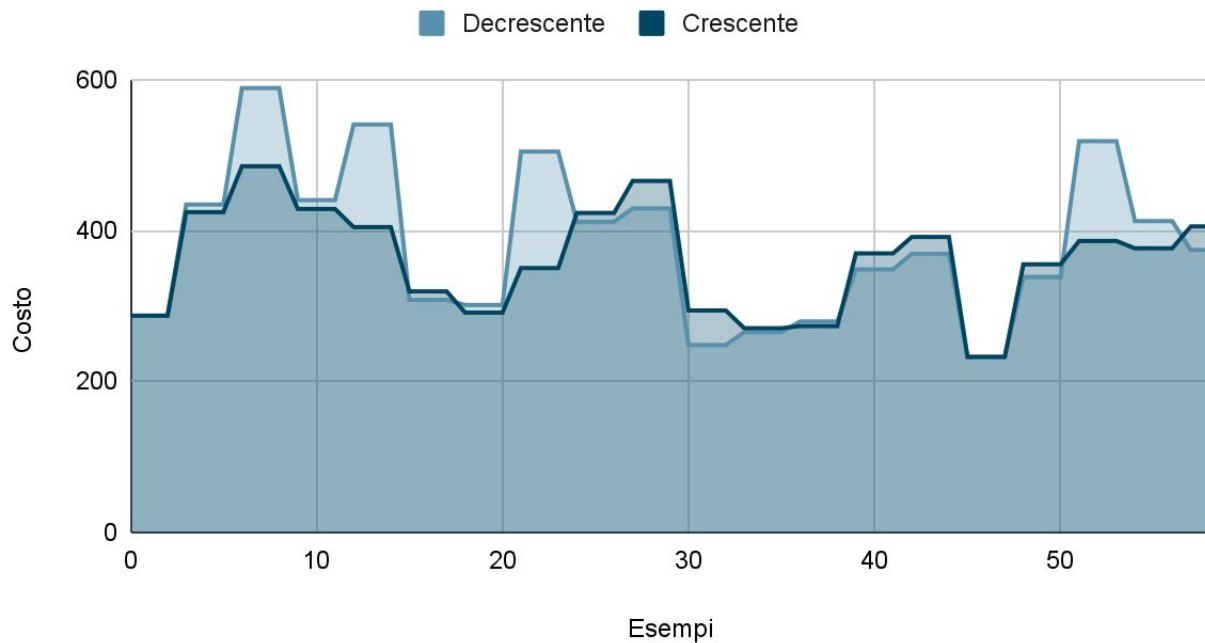
L'ordinamento crescente in media funziona meglio.

Avg:

Decrescente: 375.716

Crescente: 353.404

Decrescente e Crescente





LocalSearch

- Partendo dall euristica costruttiva precedente, l'algoritmo effettua N swap cercando di generare una soluzione migliore fino a che non scatta il timeout, non finiscono le iterazioni disponibili 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 e le iterazione non vengono considerate in questo test.

Avg:

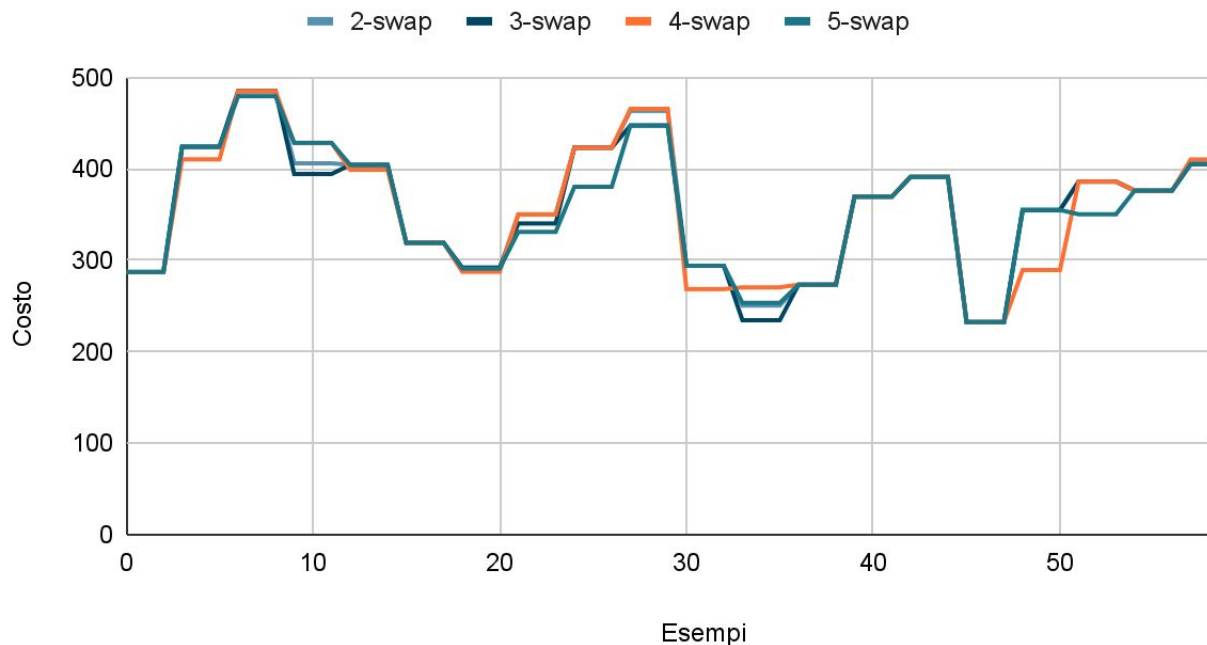
2-swap: 351.021 - 15ms

3-swap: 349.006 - 16ms

4-swap: 347.652 - 20ms

5-swap: 346.288 - 22ms

Confronto swap





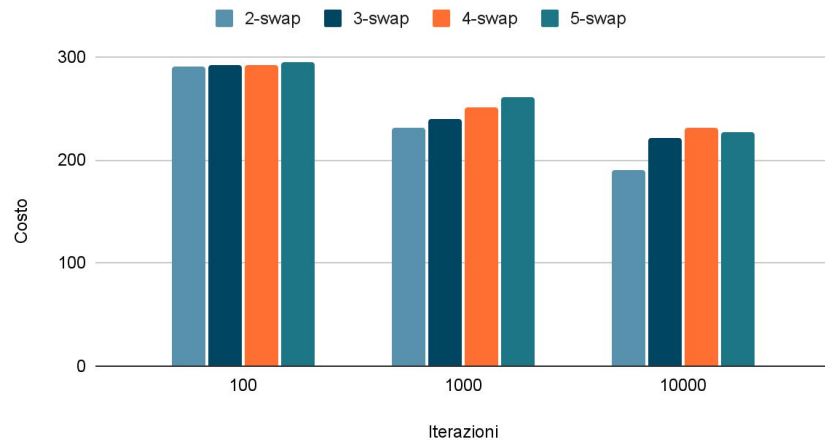
DepthLocalSearch algorithm

- 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.

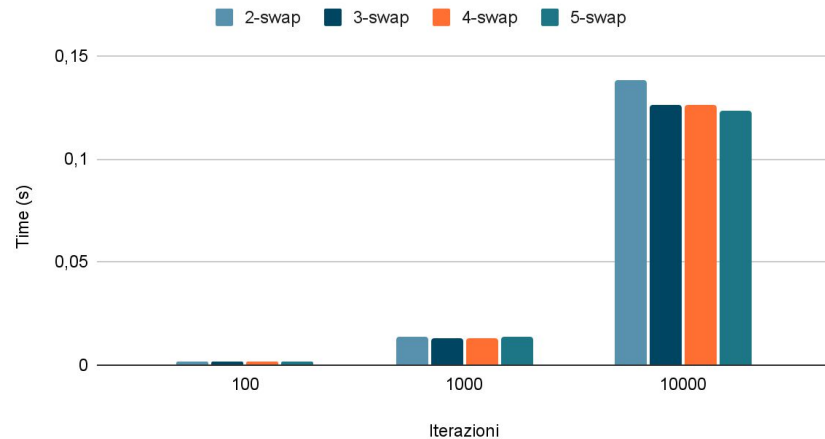
Result - DepthLocalSearch

Confronto sulla base del numero
di swap e delle iterazioni

Confronto costo



Confronto tempo





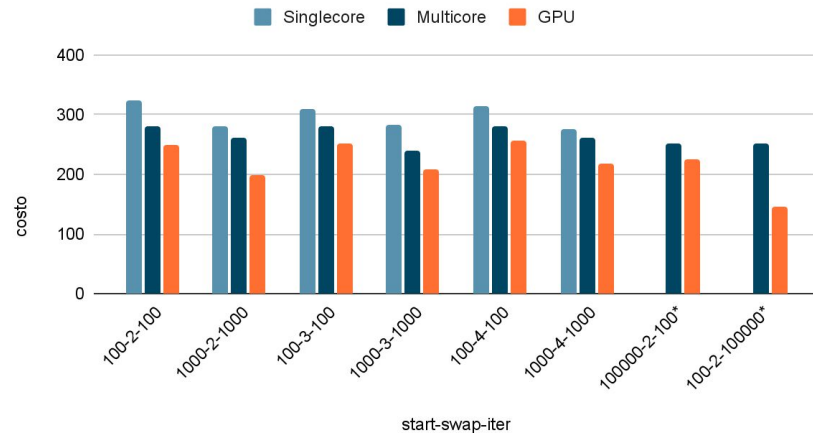
Multistart algorithm

- 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.

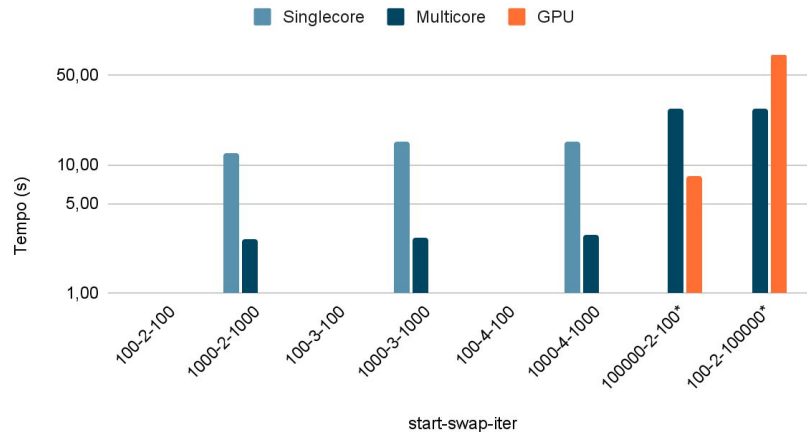
Result - Multistart

Confronto sulla base del numero di start, swap e iterazioni

Confronto costi



Confronto tempi



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



Simulated Annealing algorithm

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

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

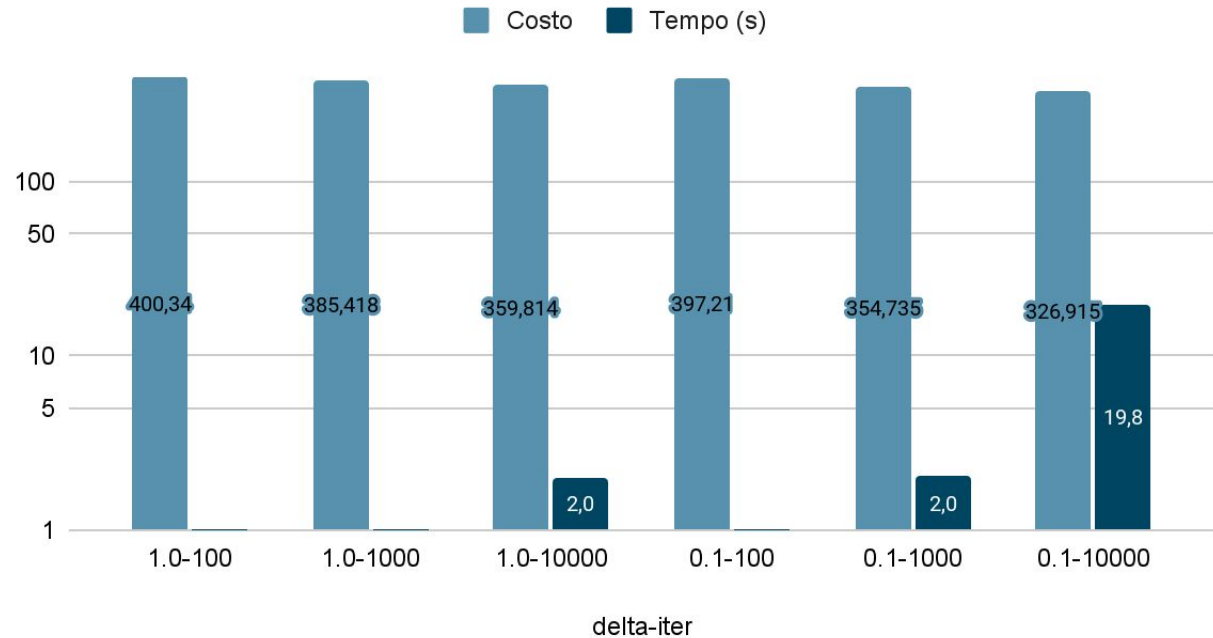
→ Parametri:

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

Result - Simulated Annealing

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

Confronto Simulated Annealing





TabuSearch algorithm

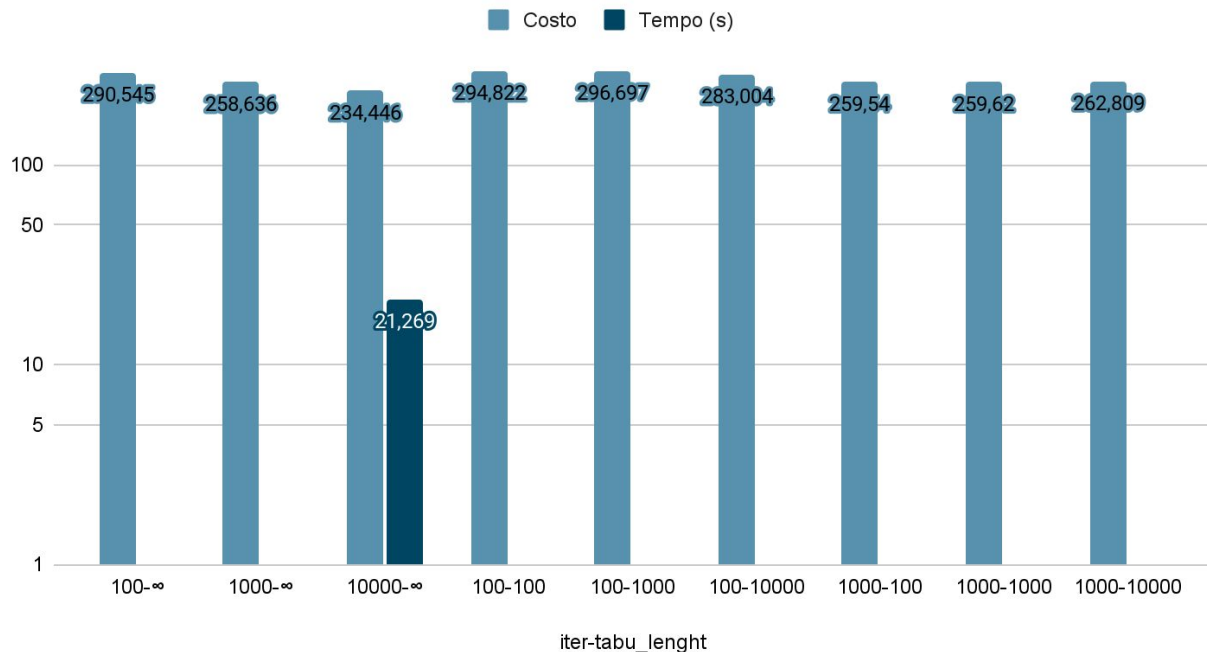
- Partendo da una soluzione iniziale tramite euristica costruttiva, l'algoritmo effettua N_{swap} per N_{it} iterazioni prendendo anche soluzioni che si discostano di un certo threshold per scappare da minimi locali. Soluzioni già esplorate vengono evitate tramite l'inserimento in una tabu list con lunghezza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

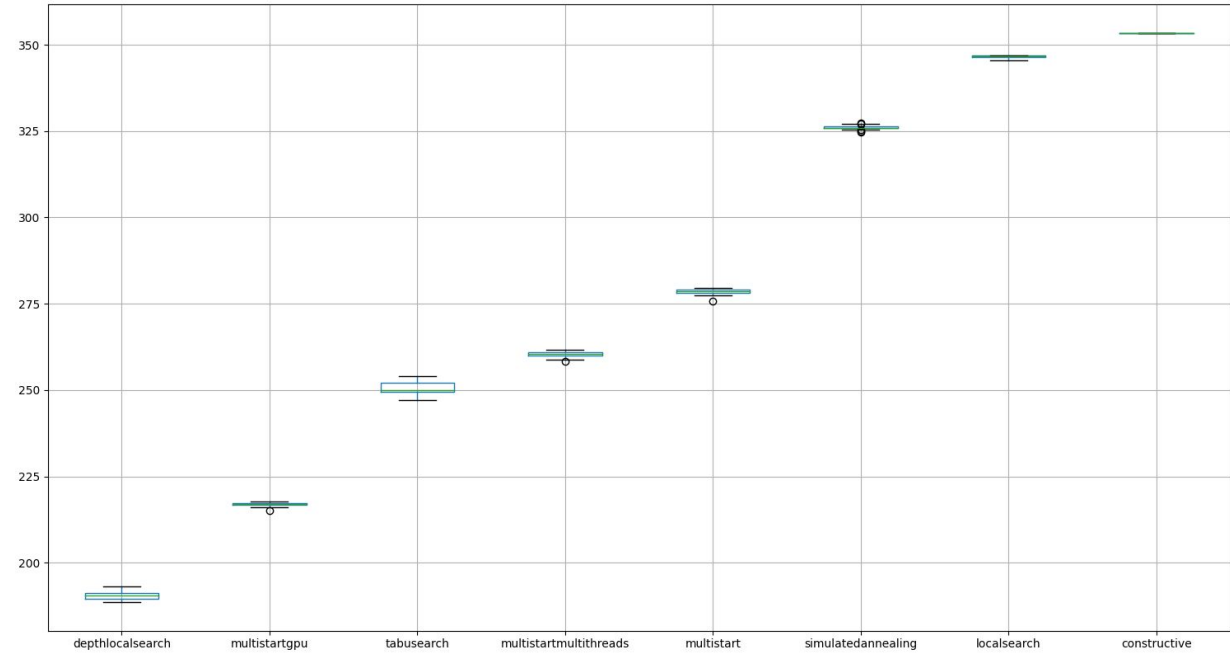
Result - TabuSearch

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

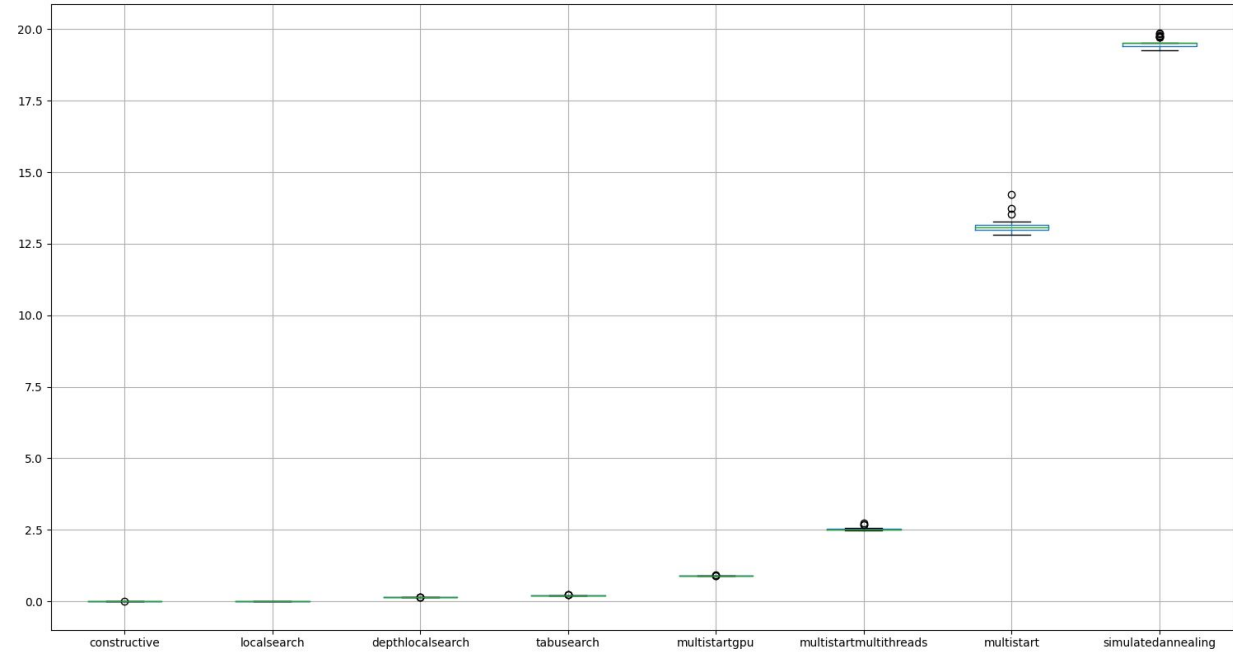
Confronto Tabu List



Confronto Costo



Confronto Tempi



Classifica Valutazione

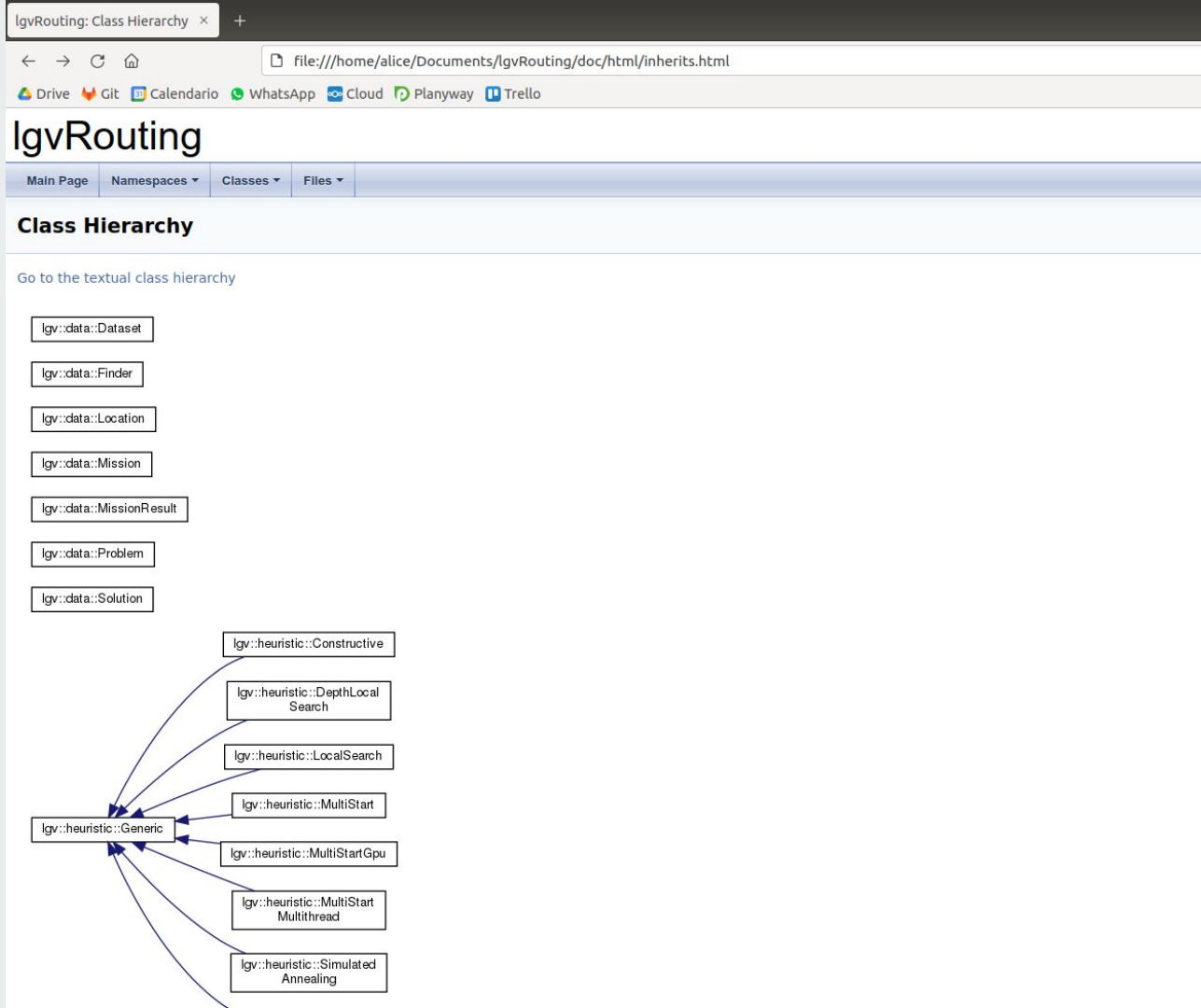
Si stila una classifica in base a $AVG_{\text{time}} / AVG_{\text{cost}}$

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

Codice

- Documentazione con Doxygen
- Test con catch2

<https://github.com/lucabart97/IgvRouting>



Grazie dell'attenzione

Luca Bartoli

228618@studenti.unimore.it

Massimiliano Bosi

205839@studenti.unimore.it

0	50	50	0 9999 9999 9999	0	0	0	(depot)
1	5	5	0 9999 9999 9999	10	0	0	(stop 1)
	
100	95	95	0 9999 9999 9999	10	0	0	(stop 100)