LA RETE DEI TRASPORTI PUBBLICI

Membri: Alberto Gallinaro, Alberto Bezzon, Tommaso Carraro

Domanda 1

Per risolvere il problema abbiamo creato un grafo orientato pesato, come richiesto dalla consegna. I nodi del grafo rappresentano le stazioni. Ogni nodo è rappresentato tramite la seguente struttura dati (dizionario python):

- identificativo univoco della stazione;
- nome della stazione;
- coordinate geografiche della stazione;
- lista degli archi uscenti dal nodo: ogni elemento della lista è rappresentato tramite un dizionario python, contenente le seguenti informazioni:
    - identificativo della stazione di arrivo;
    - orario di partenza dal nodo;
    - orario di arrivo nel nodo collegato;
    - identificativo della linea che connette i due nodi.

Ogni arco del grafo rappresenta quindi una tratta di un mezzo di trasporto (bus, treno) da una stazione ad un'altra.

Nel nostro caso il grafo presenta archi paralleli tra due nodi, questo è dovuto al fatto che due stazioni possono essere collegate attraverso diversi mezzi di trasporto, ognuno dei quali ha molteplici corse verso la stessa stazione (ogni corsa quindi è un arco a sè stante).

Ad esempio Padova e Bologna sono collegate con un autobus flixbus con tre corse al giorno che partono alle: 8.30, 12.30 e 17.30. Nel nostro caso queste tre corse saranno tre archi che partono dal nodo Padova e arrivano al nodo Bologna.

Domanda 2

L'algoritmo che abbiamo deciso di utilizzare è **A***, in quanto ideale per trovare il cammino minimo da un nodo sorgente a un nodo destinazione. L'algoritmo A* è un'astrazione del più famoso algoritmo di Dijkstra ed utilizza una funzione euristica per determinare il costo di un cammino.

L'euristica che abbiamo individuato è la seguente: "tempo di percorrenza della distanza in linea d'aria tra uno specifico nodo ed il nodo goal".

La funzione euristica affinchè funzioni correttamente deve essere ammissibile, ovvero non deve mai sovrastimare il costo per arrivare ad un determinata stazione. Per garantire tale proprietà la distanza in linea d'aria non viene percorsa con un mezzo di trasporto qualunque ma con quello più veloce disponibile in Lussemburgo, ovvero un treno che viaggia a **320km/s**.

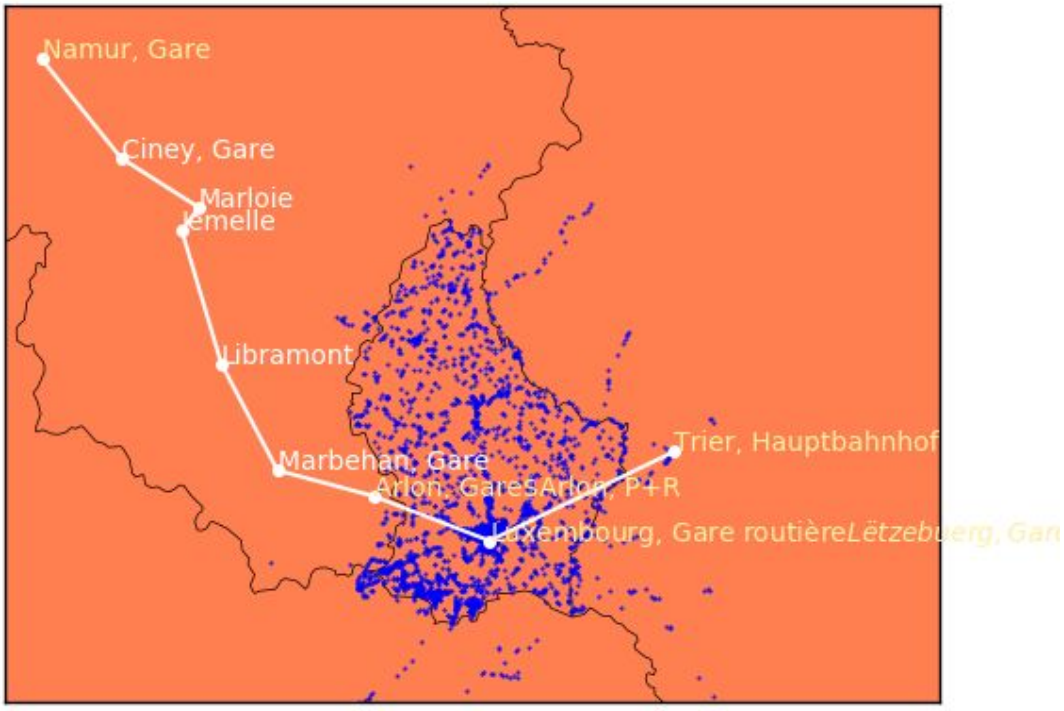
L'algoritmo originale presenta le seguenti modifiche:

- l'algoritmo tiene conto del fatto che non è possibile prendere una tratta prima del tempo di arrivo alla stazione corrispondente. Inoltre permette di rispettare le preferenze dell'utente (es: partenza non prima delle 9.30). Ad esempio se si arriva

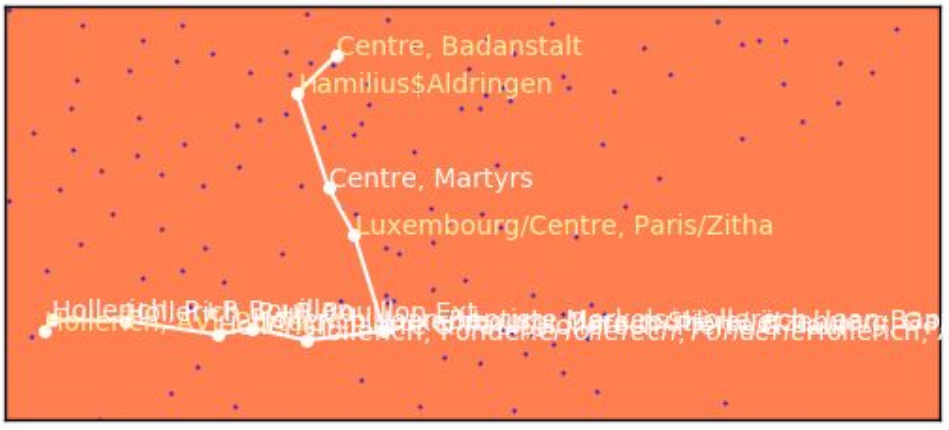
alla stazione di Padova alle 9.45 non è possibile prendere una tratta (bus, treno) che parte prima delle 9.45;

- supponendo che un viaggio non possa durare più di due giorni, date le dimensioni ristrette del Lussemburgo, per trovare la soluzione ottima, l'algoritmo tiene conto di tutte le tratte che partono da una stazione sia durante la giornata corrente che quella successiva (somma di 24 ore a tutti gli orari delle tratte). Questo perchè potrebbe verificarsi che la tratta migliore parta il giorno successivo. Ad esempio, supponendo di partire alle 8.30, posso decidere di prendere un treno alle 8.40 che giunge a destinazione alle 8.20 del giorno dopo, oppure aspettare il treno delle 8.10 del giorno dopo che mi porta a destinazione alle 8.15 (soluzione ottima).
  Inoltre, poiché il costo provvisorio della soluzione ottima è dato da: tempo di arrivo al nodo corrente (in minuti) + valore dell'euristica, in minuti, dal nodo corrente al nodo goal, il fatto di tenere conto del giorno dopo permette di risolvere il seguente problema:
  supponiamo di dover prendere un treno dopo le 23:55 e che la prima tratta disponibile sia alle 00:07.
  Se non tenessimo in considerazione le tratte del giorno successivo, l'algoritmo non sarebbe in grado di fornire una soluzione, in quanto per assicurarci che una tratta parta dopo l'orario di arrivo in una stazione, gli orari vengono convertiti in minuti (vincolo dettato dall'euristica). Quindi l'orario 00:07 convertito in minuti è 7 che è inferiore rispetto a 23:55 convertito in minuti, di conseguenza l'algoritmo non sarebbe in grado di individuare la tratta 00:07, nonostante questa possa essere presa.
  La nostra soluzione risolve il problema, in quanto tiene conto anche dell'orario 24:07 (in minuti maggiore di 23:55) che sarebbe la tratta che parte alle 00:07 ma del giorno successivo.
  Questo è stato implementato tramite due cicli sugli archi uscenti da ciascun nodo. Il primo ciclo trova la soluzione ottima per il nodo al giorno 1, mentre il secondo ciclo verifica se esiste una soluzione migliore tra le corse al giorno successivo (tiene conto di tutti gli orari incrementati di 24 ore).

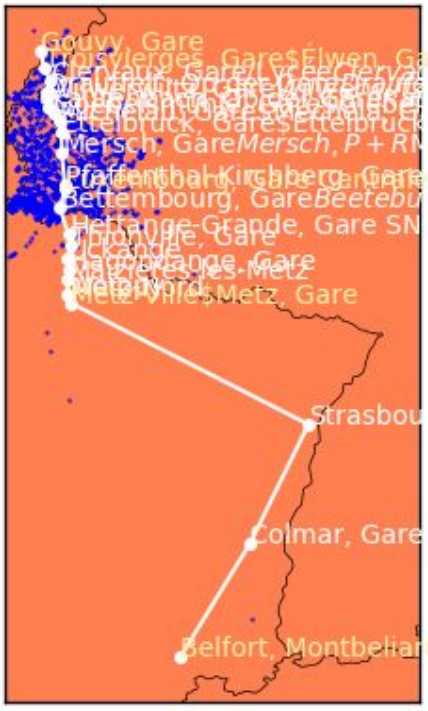
Domanda 3
Viaggio da 500000079 a 300000044
Orario di partenza 13:00
Orario di arrivo 17:18
13:46 : corsa 06171 CFLBUS da 500000079 a 200405036
14:47 : corsa 06311 CFLBUS da 200405036 a 300000003
15:31 : corsa 02138 C82--- da 300000003 a 300000044



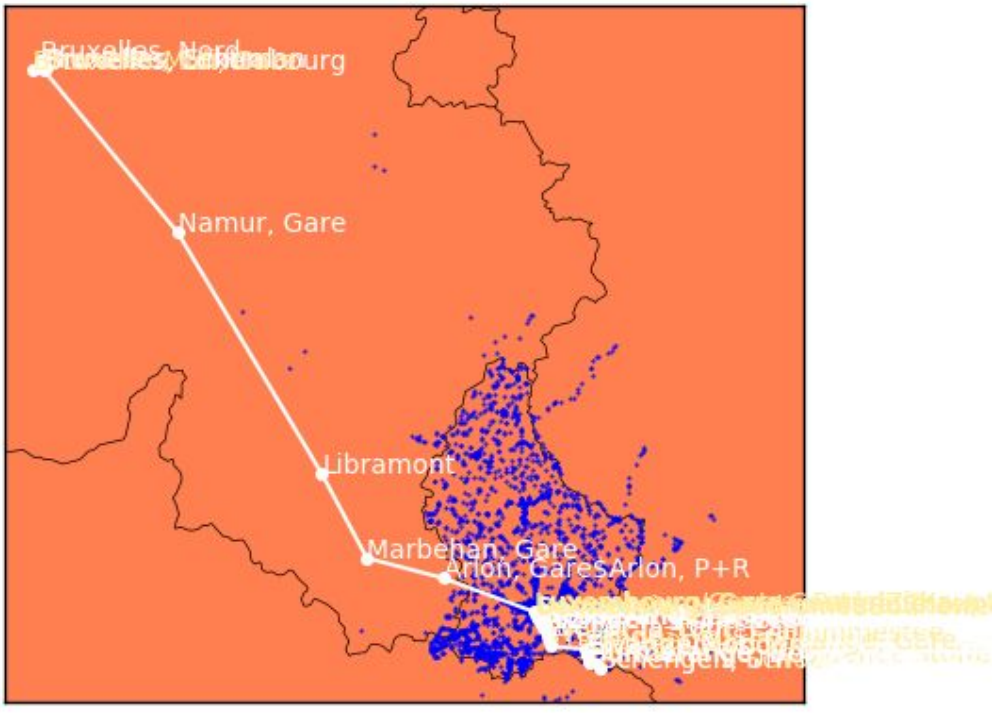
Viaggio da 200415016 a 200405005
Orario di partenza 9:30
Orario di arrivo 9:52
9:30 : corsa 00360 RGTR-- da 200415016 a 200405026
9:40 : corsa 01797 AVL--- da 200405026 a 200405020
9:50 : corsa 06602 RGTR-- da 200405020 a 200405005



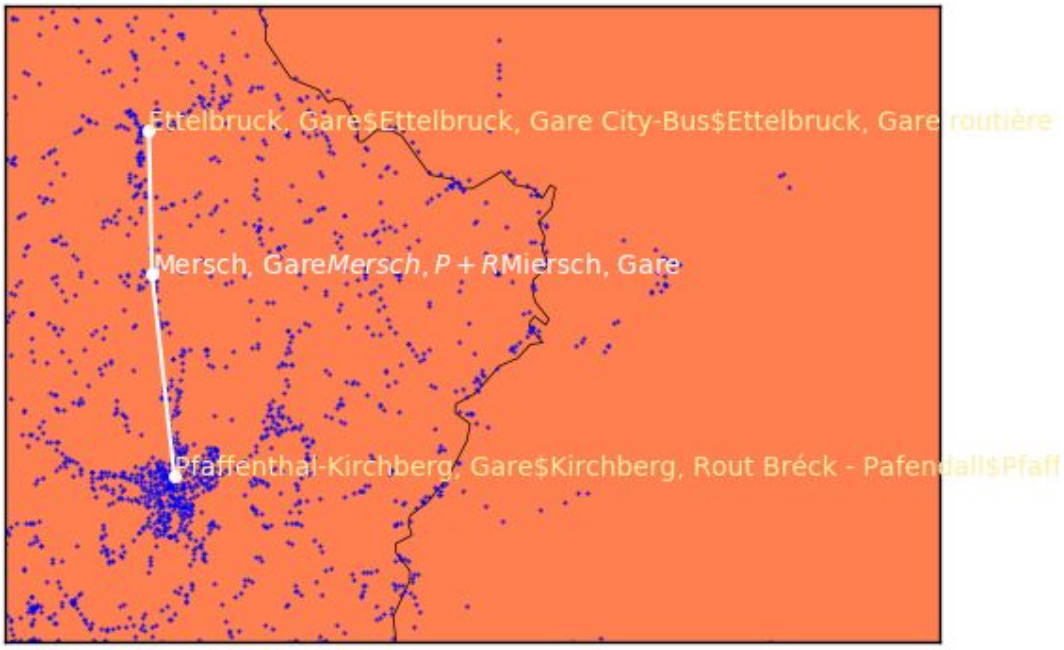
Viaggio da 300000032 a 400000122
Orario di partenza 5:30
Orario di arrivo 13:50
6:26 : corsa 07608 C88--- da 300000032 a 110606001
6:35 : corsa 03781 C82--- da 110606001 a 200405035
7:46 : corsa 00055 C82--- da 200405035 a 400000047
12:07 : corsa 09879 C82--- da 400000047 a 400000122



Viaggio da 210602003 a 300000030
Orario di partenza 6:30
Orario di arrivo 10:53
6:41 : corsa 00030 CFLBUS da 210602003 a 210201002
6:46 : corsa 00036 CFLBUS da 210201002 a 210502001
6:55 : corsa 00037 CFLBUS da 210502001 a 201103004
7:07 : corsa 01306 CFLBUS da 201103004 a 200404016
7:20 : corsa 00031 CFLBUS da 200404016 a 200405036
7:24 : corsa 01173 RGTR-- da 200405036 a 200405026
7:27 : corsa 04278 AVL--- da 200405026 a 200405035
7:40 : corsa 07630 C82--- da 200405035 a 300000030



Viaggio da 200417051 a 140701016
Orario di partenza 12:00
Orario di arrivo 12:43
12:20 : corsa 03762 C82--- da 200417051 a 140701016

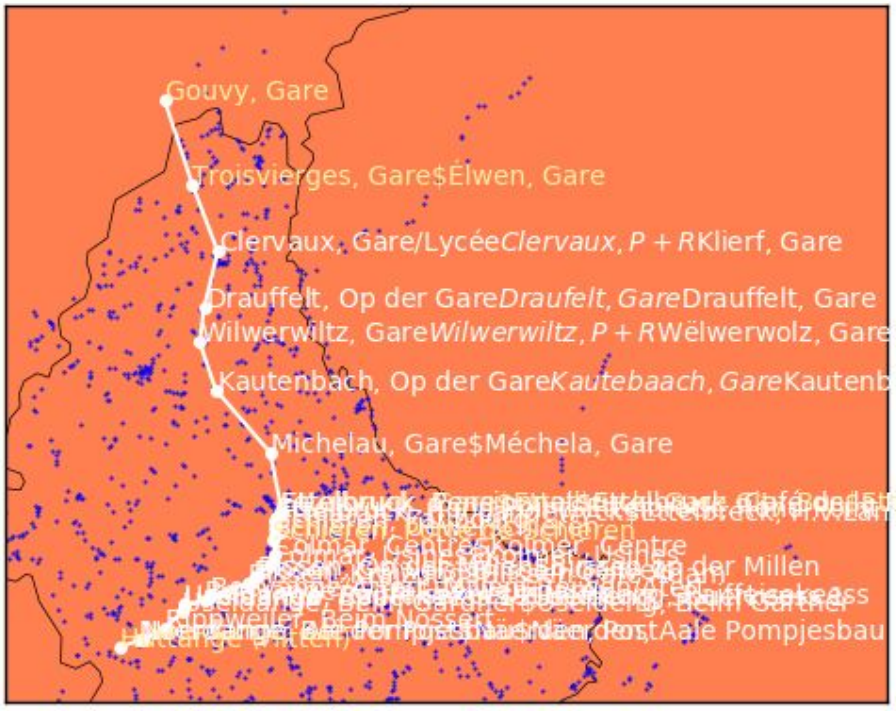


Viaggio da 200417051 a 140701016
Orario di partenza 23:55
Orario di arrivo 24:44
24:09 : corsa 03623 C82--- da 200417051 a 140701016

Viaggio da 120904001 a 120103002
Orario di partenza 10:55
Orario di arrivo 12:24
11:08 : corsa 02002 RGTR-- da 120904001 a 121007001
11:36 : corsa 00850 RGTR-- da 121007001 a 120903038
12:09 : corsa 06174 RGTR-- da 120903038 a 120707001
12:14 : corsa 06176 RGTR-- da 120707001 a 120103002

Viaggio da 150303001 a 130107002
Orario di partenza 7:55
Orario di arrivo 10:42
8:04 : corsa 03059 RGTR-- da 150303001 a 150606002
8:21 : corsa 03321 RGTR-- da 150606002 a 140801002
8:58 : corsa 01750 RGTR-- da 140801002 a 120503001
9:32 : corsa 07628 RGTR-- da 120503001 a 120903021
10:04 : corsa 02075 RGTR-- da 120903021 a 120902001
10:13 : corsa 07734 RGTR-- da 120902001 a 130107002

Viaggio da 300000032 a 150104001
Orario di partenza 3:20
Orario di arrivo 6:51
5:26 : corsa 07606 C88--- da 300000032 a 110606001
5:35 : corsa 03780 C82--- da 110606001 a 140701016
6:15 : corsa 07474 RGTR-- da 140701016 a 141301007
6:25 : corsa 06016 RGTR-- da 141301007 a 150104001

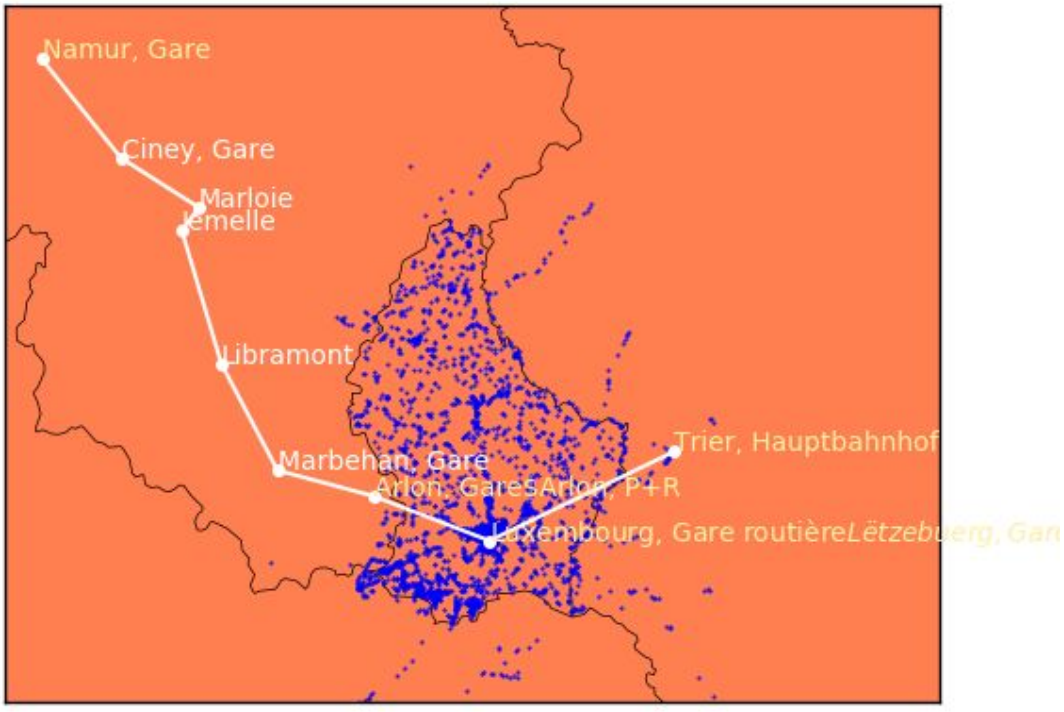


Domanda 4
Riteniamo che la soluzione fornita dalla nostra implementazione sia corretta e ottimale, in quanto in linea con le soluzioni fornite da Google Maps, sia per tempi di percorrenza che per tratte percorse. Viene di seguito fornito un esempio:

Viaggio da 500000079 (Trier) a 300000044 (Namur)

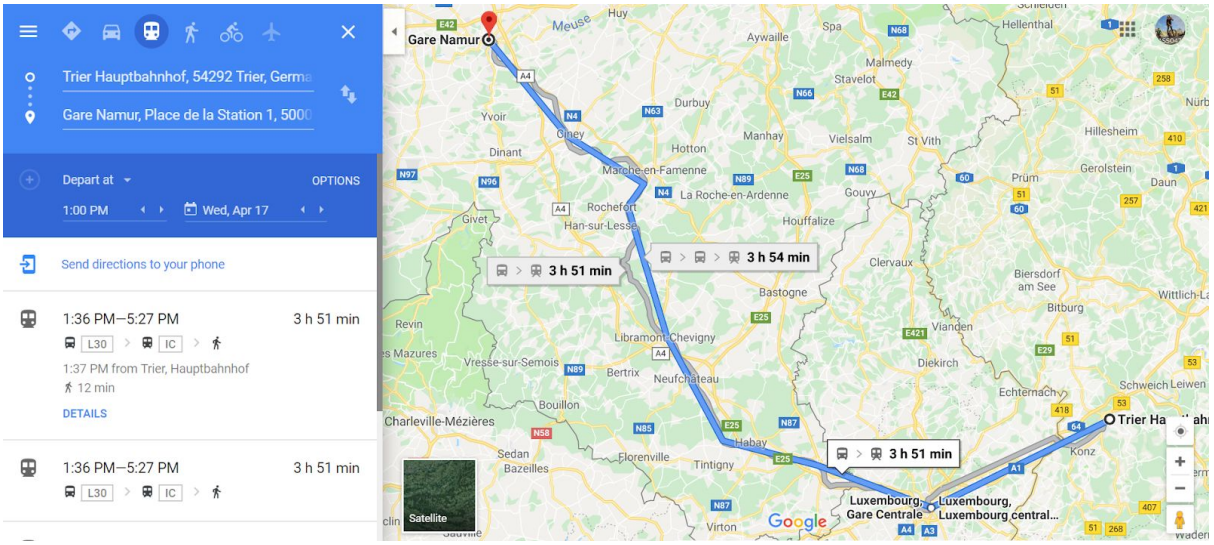
Nostra soluzione

Orario di partenza 13:00
Orario di arrivo 17:18
13:46 : corsa 06171 CFLBUS da 500000079 a 200405036
14:47 : corsa 06311 CFLBUS da 200405036 a 300000003
15:31 : corsa 02138 C82--- da 300000003 a 300000044



Soluzione di Maps

Tempo di percorrenza: 3 ore e 51 minuti
Orario di partenza: 13:00
Orario di arrivo: 17:27
L'orario è in linea ma non identico, sia per il fatto che il dataset potrebbe non essere aggiornato, sia per il fatto che la soluzione di maps parte alle 13.36 invece che alle 13:46.



In conclusione, abbiamo confrontato la nostra soluzione con quelle degli altri gruppi ed i risultati collimano ma generalmente le nostre soluzioni sono quelle che comportano meno cambi di linea, quindi migliori per l'utente finale.