Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica Magistrale

Analisi di Politiche di Car Pooling basata su tracce GPS

Relatore:
Chiar.mo Prof.
Luciano Bononi

Correlatore:
Dott. Luca Bedogni

Presentata da:
Antonio Carbonara

Sessione II
Anno Accademico 2016/2017

Alla mia famiglia e a tutti coloro
che mi sono stati vicino, sempre.

Sommario

Il problema della mobilità veicolare è un tema al quale la comunità di ricerca ed in generale le amministrazioni cittadine sono molto sensibili.
A riguardo sono stati proposti vari studi uniti alla rielaborazione di dati *raw*, i quali si riferiscono a tracce mobili di veicoli circolanti in un'area urbana.
Il contributo da noi offerto consiste proprio nella proposizione di uno strumento in grado di rielaborare tracce GPS e di realizzare, a partire da queste tracce di mobilità rielaborate, aggregazioni (in termine tecnico *cluster*) di veicoli al fine di studiare la riduzione di veicoli all'interno di una qualsiasi area urbana di cui si dispone dei rilevamenti GPS con conseguenti vantaggi che si potrebbero ottenere sulla qualità di vita dei cittadini e delle città stesse.

Introduzione

Oggigiorno il caro carburante rappresenta un problema che pone dei limiti alla qualità di vita dei cittadini sia a livello lavorativo che sociale.
Per cercare di limitare questo problema sempre più crescente diventa quindi di primaria importanza la possibilità che un cittadino possa condividere la propria auto con altre persone, condividendo e riducendo le spese del carburante oppure usufruire di mezzi messi a disposizione dai servizi cittadini, posizionati in vari punti di una città.

I servizi in questione possono essere di due tipi: *car sharing* oppure *car pooling*. Nel primo caso, si fa riferimento al termine *Car Sharing* ([14]) intendendo un modello di mobilità urbana in cui un fruitore del servizio noleggia un'auto e paga il servizio in base all'utilizzo che ne ha fatto. È chiaro che un servizio del genere potrebbe comportare una serie di vantaggi a livello economico da parte del guidatore che ne usufruisce e l'auto utilizzata potrebbe essere riutilizzata da un altro utente subito dopo che è stata lasciata libera dall'utente precedente.

Con il termine *Car Pooling* ([13]), invece, ci si riferisce ad un servizio differente dal precedente in cui un guidatore mette a disposizione la propria auto offrendo un passaggio ad altre persone che devono raggiungere una destinazione non molto distante da quella del guidatore, condividendo i costi del viaggio. In questo caso i vantaggi per tutti gli utenti in questione possono spaziare dal risparmio sui costi del carburante, olio, costi di parcheggio, pedaggi, ecc. alla riduzione generale delle auto circolanti in una città con conseguenti vantaggi ad impatto ambientale e quindi riduzione dell'inquinamento

nella città.

Tali tematiche hanno interessato anche le comunità di ricerca informatiche volte a studiare e realizzare applicativi aventi come fine quello di incentivare gli utenti ad usufruire dei servizi descritti poc'anzi. Tramite questo lavoro di tesi, di fatti, si è voluto analizzare il dominio della mobilità veicolare realizzando un *tool chain* scritto in *python* volto ad incentivare politiche di *Car Pooling* a partire da tracce GPS contenute in un dataset di file *.csv* che saranno descritti in seguito.

Ci addentriamo ora nella descrizione del lavoro di tesi realizzato e articolato nei seguenti capitoli:

- a questa Introduzione seguirà il Capitolo denominato *Stato dell'Arte* in cui vedremo gli argomenti direttamente collegati al nostro studio, i quali spaziano dall'analisi di grandi dataset, alla gestione di dati *raw* ed al problema della mobilità veicolare
- Il Capitolo 2 che porta il titolo di *Proposta* è volto a descrivere l'implementazione del *tool chain* realizzato per effettuare l'analisi di politiche di *car pooling* basata su tracce GPS
- il Capitolo 3 mostra i risultati ottenuti dagli studi avviati dal *tool chain*
- infine riporteremo nelle *Conclusioni*, le considerazioni sul lavoro realizzato ed i possibili lavori futuri al quale uno strumento del genere potrebbe contribuire

Indice

Capitolo 1

Stato dell'Arte

In questo capitolo vedremo quello che è lo stato dell'arte attuale ed i temi direttamente correlati all'analisi di politiche di **car pooling** basata su tracce GPS, realizzata ed illustrata in questa tesi.

1.1 Analisi di grandi dataset

Un tema particolarmente delicato ed importante del quale si occupano i ricercatori informatici, oltre che altri esperti quali ad esempio gli operatori di mercato oggigiorno, riguarda il fatto di fornire strumenti per l'analisi di grandi dataset di dati. Di fatti come si può vedere in [8], Furno *et al.* hanno proposto un approccio originale basato su studi statistici e modelli matematici in grado di analizzare un grosso dataset contenente dati di traffico mobile (che nel 2015 constava di 3.7 exabytes di dati) ed inferire strutture spaziali e temporali nella richiesta del traffico mobile (altresí detto *mobile traffic demand*). Lo scopo dello studio da loro condotto è stato quello di risolvere due problemi legati all'analisi del traffico mobile:

- **Network activity profiling**, il quale permette di classificare insieme periodi di tempo che mostrano una distribuzione spaziale stabile nella *mobile traffic demand*
- **Land use detection**, il quale si occupa di identificare strutture spaziali nei dati di traffico mobile, tramite la decomposizione di un'area ge-

ografica in zone in cui le dinamiche di traffico mobile sono eterogenee nel tempo

Quindi questi due problemi sono stati risolti con tecniche, come detto, matematiche e statistiche appartenenti ad un tipo di approccio denominato *Exploratory Factor Analysis* (EFA), il quale pur essendo applicato ad un enorme dataset di dati mobili, ha rivelato la presenza di caratteristiche quali ad esempio la presenza di una forte periodicità temporale ed una forte località geografica le quali permettono di prevedere la richiesta del traffico mobile.

In un *survey* proposto da Naboulsi *et al.* ([10]) viene sottolineata l'importanza di questo argomento in quanto viene dimostrato che analizzando enormi dataset contenenti dati mobili generati dai vari utenti usufruenti dei servizi mobili, è possibile studiare fenomeni legati al campo della finanza, sociologia, epidemologia, urbanistica, ecc. e quindi progettare possibili soluzioni ai problemi che si presentano. Ad esempio se si rimane concentrati sul problema della mobilità veicolare e delle possibili soluzioni che possono essere proposte dall'applicazione di politiche di *car-sharing* o di *car pooling*, in uno studio del *survey* descritto in [10] sulla base di approcci stocastici è stato possibile costruire modelli di mobilità in relazione alla distribuzione empirica delle celle di operatori mobili alle quali si sono agganciati gli utenti che hanno prodotto un qualsiasi dato mobile (sia esso un dato di tipo voce, testo o che fa riferimento a servizi come ad esempio quelli offerti da Google Maps) ed è stato possibile mappare il traffico mobile generato direttamente sui dati della rete stradale ottenendo dei risultati plausibili. Quindi è stato dimostrato che per una città come Madrid è possibile studiare le tracce mobili per applicare casi di *car sharing* ottenendo una riduzione di circa il 67% delle auto con gli autisti disposti ad accettare deviazioni di al massimo 600 metri rispetto ai loro percorsi consueti. Inoltre in una città come Shenzen, paragonando le tracce generate dal traffico mobile con i flussi di trasporti pubblici, sono state proposte nuove linee di autobus in grado di ridurre il tempo di viaggio di una giornata tipica di lavoro del 25%.

L'analisi di grandi dataset siano essi composti di dati di traffico mobile o

relativi al traffico cittadino, permette di estendere questi studi alla ricerca di pattern socio-topologici in scenari urbani che riguardano medie/grandi città. Come descritto in [7], sono state applicate metodologie matematiche al fine di mostrare l'esistenza di una forte relazione tra le comunicazioni mobili e le varie infrastrutture (e.g. strade, sistemi di trasporti, edifici per l'educazione, lo sport, la salute, commerciali, industriali, ecc.), le quali caratterizzano differenti zone di un'area metropolitana. Tali correlazioni sono state scoperte all'interno di un generico dataset contenente attività di comunicazioni mobili di un certo campione di popolazione, comunicazioni che si sono svolte durante un certo insieme di giorni suddivisi a loro volta in intervalli di tempo (*time slots*). Per ogni giorno i dati mobili prodotti da ogni utente sono stati aggregati in insiemi di dati che condividono la stessa area e quindi in questo modo è possibile variare, in fase di analisi, la granularità spaziale e temporale del dataset analizzato. Questo approccio ha permesso di analizzare città come Milano e Torino, ma anche Lione, Parigi, Marsiglia o Tolosa, suddividendole in varie zone in base alle aree di copertura delle celle nelle quali sono state rilevate attività mobili e grazie alle metodologie descritte in [7] è stato possibile rivelare una forte correlazione tra le attività mobili e le strutture entro le quali le tracce mobili sono state prodotte, così da rivelare anche le abitudini di una popolazione di una nazione o, come in questo caso, di una città. Ad esempio, tramite questo studio, è possibile comprendere se in una città sono più frequentate le attività commerciali rispetto a quelle turistico-culturali.

1.2 Il problema della mobilità veicolare

Il problema della mobilità veicolare, che in fondo è il tema principale di questa tesi, è una tematica cruciale nell'organizzazione delle città di gran parte del mondo. Questa grazie al sussidio ed al perfezionarsi delle tecnologie informatiche, viene sempre più presa in considerazione dagli esperti direttamente presenti nei vari comuni od anche da ricercatori accademici al fine di migliorare la qualità dei servizi di trasporto o dei flussi di traffico dei centri cittadini. Ad esempio, sono stati proposti degli studi di simulazione ([9]) su reti veicolari al fine di comprenderne le caratteristiche topologiche di queste

reti istantanee, così da poter estendere l'applicazione di tali studi in casi reali che possono comprendere: lo studio di nuovi protocolli di rete o di nuovi software da poter fornire alle aziende o ai comuni intenzionati a migliorare la qualità dei sistemi di trasporto.
Data la mancanza di dataset di riferimento, Bedogni *et al.* ([1]) hanno proposto un dataset per la mobilità veicolare basato su dati di traffico verosimili della città di Bologna ed ideato tramite modifiche e migliorie apportate ai tool forniti da SUMO (Simulation of Urban MObility) e OSM (OpenStreetMap), il quale è stato successivamente validato dai servizi offerti da Google Maps API. Un altro contributo fornito per quanto riguardo studi per il problema della mobilità veicolare, ha riguardato il fatto di realizzare un dataset di tracce di mobilità a partire da Open Data prendendo come caso di studi sempre la città di Bologna ([3]). In particolare, grazie a questo lavoro è stata sottolineata la potenzialità fornita dagli Open Data nel creare modelli verosimili di mobilità urbana. A partire da questi dati resi pubblicamente accessibili, sono state sviluppate delle *Origin-Destination Matrices* (ODM) tramite le quali è stato possibile descrivere dei pattern di viaggio mostrando, appunto, il numero di viaggi tra differenti zone dell'area urbana della città di Bologna (che si estende per circa 28 km^2) durante un'intera giornata (24 ore).
Inoltre sono stati avanzati casi di studio reali con soggetto il *car sharing* ([5], [6]) aventi come scopo quello di comprendere in base a quali aspetti comportamentali, determinati utenti sono disposti ad accettare un servizio del genere.

Leggermente differente dal *car sharing*, è il tema del *car pooling* il quale ha attratto l'attenzione in ambiti di ricerca accademici, in quanto rappresenta un argomento che sta diventando di interesse giornaliero poichè potrebbe migliorare la qualità di vita dei lavoratori ed in generale di tutte quelle città con elevato tasso di inquinamento.
Di fatti Bruglieri *et al.* ([2]) hanno presentato un servizio di *car pooling* per l'Università Statale ed il Politecnico di Milano, sviluppando un algoritmo con tecniche di *social network analysis* al fine di incentivare l'utilizzo del

car pooling tra studenti ed impiegati dell'università. Inoltre, come si vede in [11], è stato realizzato uno studio comparativo tra criteri di *car pooling acceptance* (ovvero quei criteri secondo i quali un guidatore potrebbe condividere la propria auto con altre persone) tramite l'ausilio di un framework scritto in Java, mettendo a confronto un algoritmo basato sulla tecnica di *hill climbing*, appartenente al campo dell'intelligenza artificiale, ed un algoritmo *greedy* avente come scopo quello di analizzare le performance di questi algoritmi così da comprendere come migliorarli, in maniera tale da far sì che si avvicinino agli *acceptance criteria* del mondo reale.

1.3 Gestione di dati GPS grezzi

Un altro aspetto molto importante di questa tesi è certamente l'elaborazione e gestione di tracce GPS *raw*, in quanto, come vedremo, è ciò di cui si occupa inizialmente lo stumento fornito tramite il nostro contributo.
Allo stato dell'arte attuale, tale argomento è stato trattato da Celes *et al.* in [4]: lo scopo di tale lavoro è stato proteso a colmare il gap esistente tra dei dataset di tracce di mobilità reali ed i file utilizzati per la simulazione di protocolli per le reti veicolari (VANET), cosicché da poter sfruttare le nuove tracce generate per qualsiasi lavoro futuro sia esso direttamente correlato con le VANET o con altri studi di mobilità veicolare, come ad esempio quelli mostrati nella sezione precedente. A partire dai dati GPS rilevati da veicoli provvisti, appunto, di dispositivi GPS nelle città di Colonia, Roma, Shangai, Pechino e Shenzen, i ricercatori contribuenti a [4] hanno proposto un algoritmo in grado di *clusterizzare* i punti GPS come locazioni attraversate dai veicoli ed individuare quindi la traiettoria tracciata da ciascun veicolo. In seguito è stato ideato un metodo di calibrazione avente come scopo quello di trasdurre questi dati grezzi in tracce di mobilità di grana fine. Infine tale metodo è stato validato mettendo a confronto la curva tracciata dai risultati della calibrazione con quella ricavata dai dati originali; i risultati ottenuti sono molto interessanti ed infatti Celes *et al.* hanno deciso di rendere pub-

bliche le tracce di mobilità ricalibrate in maniera tale da contribuire alla comunità di ricerca.

Tutti gli studi ed i lavori visti fino a questo momento, potrebbero aprire la strada a nuove prospettive riguardanti la tematica del *car pooling* od altri temi direttamente ad esso collegati, così come lo strumento realizzato in questa tesi e che vedremo nel capitolo successivo.

Capitolo 2

Proposta

In questo capitolo descriveremo il *tool chain* realizzato per la gestione e rielaborazione di dati GPS avente come obiettivo primario quello di *clusterizzare* tali dati, così da poter applicare analisi nel dominio del *car pooling*, volte in particolare a studiarne la possibile riduzione delle auto all'interno di una città. Prima di addentrarci nella descrizione del *tool* da noi proposto, descriveremo il dataset utilizzato e dal quale abbiamo sviluppato i nostri studi.

2.1 Il Dataset

Il dataset utilizzato per questo lavoro di tesi, è un dataset di coordinate GPS ricalibrato a partire da dati grezzi tramite interpolazioni di punti in maniera tale da poter "riempire i buchi" presenti in origine nei percorsi evidenziati nel dataset, anche tenendo a mente gli studi messi in pratica da [4] come illustrato nel capitolo prima. Le coordinate GPS sono state raccolte grazie all'ausilio di automobili della città di Bari equipaggiate di un dispositivo GPS (che può essere anche semplicemente uno smartphone acceso durante il percorso seguito dall'auto). Tali dati sono memorizzati in semplici file con estensione *.csv* aventi come nomeclatura un identificativo indicante la traiettoria di un certo viaggio svolto nella città di Bari, viaggio che per convenzione, abbiamo considerato come un singolo veicolo al fine di

poter effettuare facilmente dei *match* che descriveremo in seguito.
In realtà, tra i dataset a disposizione vi sono anche le tracce di mobilità raccolte a Roma, Firenze, Torino, Palermo, San Francisco, Pechino e Shenzen, ma si è comunque scelto di utilizzare come caso di studio, il dataset della città di Bari in quanto rappresenta forse il dataset più piccolo a livello di quantità di dati e quindi più veloce da analizzare anche per validare lo strumento da noi realizzato e che andremo a descrivere fra un po'. I file del dataset analizzato, sono stati organizzati in cartelle, in cui ciascuna cartella indica un giorno dell'anno scelto per le successive analisi, quindi in definitiva per ogni cartella ci sono i file *.csv* dei veicoli che hanno viaggiato in un certo giorno per la città di Bari, di fatti una singola cartella ha nomeclatura "bari_out_*data*", con la *data* intesa nel formato anno-mese-giorno.
Ogni file del dataset è formato da righe in cui ogni attributo del singolo veicolo viene separato dai restanti tramite un semplice spazio e per ogni riga abbiamo:

- **id**: l'identificativo del veicolo (o viaggio)
- **timestamp**: tempo in formato UNIX, indicante i secondi durante i quali il veicolo ha attraversato un determinato punto della città
- **long**: longitudine per il quale il veicolo è passato
- **lat**: latitudine per il quale il veicolo è passato

Tenendo conto di questi parametri, sono state impostate le componenti del *tool chain* che ci stiamo accingendo ad osservare nella prossima sezione.

2.2 Tool chain per studi di Car Pooling

Lo strumento realizzato per questo lavoro di tesi è stato sviluppato nel linguaggio di programmazione *python* ed è composto da differenti moduli ciascuno con un compito preciso che possono anche essere visti in maniera indipendente, anche se per effettuare un'analisi completa è bene eseguirli in maniera sequenziale. Vediamo uno per uno ciascun componente.

2.2.1 Car Writer

Prima di poter essere propriamente utilizzate, le tracce GPS, organizzate secondo quanto visto nella sezione *Il Dataset*, hanno avuto bisogno di un filtraggio per poterle adattare ai nostri studi di analisi della riduzione del numero di auto circolanti nella città di Bari al fine di incentivare politiche di *car pooling*. Di questo filtraggio se ne occupa il modulo *car-writer.py*, il quale accetta come input il percorso di un singolo file *.csv*, rappresentante la mobilità di un singolo veicolo, e grazie alle funzioni di gestione dei file messe a disposizione dal linguaggio *python*, recupera solamente la prima e l'ultima riga del file in quanto per i nostri studi non è stato di nostro interesse tenere traccia anche dei punti di interpolazione e quindi di tutto il percorso dell'auto, ma solamente dei punti di partenza e di arrivo di ciascun veicolo. Dunque viene scritta una singola riga in un file denominato *allvehicles.csv* aperto in *append*, con i seguenti attributi separati da spazi:

- identificativo del veicolo
- tempo (timestamp) di partenza
- tempo (timestamp) di arrivo
- longitudine di partenza
- latitudine di partenza
- longitudine di arrivo
- latitudine di arrivo

Tuttavia prima di poter inserire la riga nel file *allvehicles.csv*, è stato necessario effettuare un ulteriore filtraggio, verificando che le coordinate iniziali non siano uguali a quelle finali in quanto questo indica che il veicolo in questione non ha fornito nessun contributo alla rilevazione dei dati GPS, poichè ha disattivato il dispositivo GPS e quindi tutti i veicoli che hanno presentato questa caratteristica sono stati esclusi dallo studio.

2.2.2 Car Analysis

Il *core* del *tool chain* è sicuramente rappresentato dal modulo denominato *car-analysis.py* che si occupa di realizzare i *matching* possibili tra i veicoli e di memorizzarli in dei file *.csv* appositamente creati i quali torneranno utili per i nostri studi. Si è scelto di rendere questo modulo il più generico possibile in maniera tale da astrarre dal tipo di analisi che si vuole eseguire nell'ambito del *car pooling*, di fatti *car-analysis.py* prende in input:

- `T`: un tempo espresso in secondi
- `D`: una distanza espressa in metri
- α, β: parametri utili per una **funzione obiettivo** che descriveremo in seguito
- `N`: il numero di passegeri di un veicolo
- `city_dir`: cartella del dataset dentro la quale saranno spostati i file *.csv* creati dal modulo al termine dell'analisi

Particolare importanza assumono i parametri `T`, `D` ed `N` in quanto variandoli, è possibile realizzare varie analisi e capire quali comportamenti si manifestano nella città del dataset analizzato. In particolare, `T` modellizza il *delay* (ma anche l'anticipo) che un guidatore, disposto a condividere la propria auto potrebbe accettare sia sulla partenza che in arrivo così come `D` modellizza la distanza oltre la quale il guidatore ospitante può spingersi per recuperare l'ospite da accompagnare, sia in partenza che in arrivo. Infine `N` indica, come detto, il numero di passeggeri del veicolo e nel caso del *car pooling*, ovviamente il numero massimo consentito per effettuare uno studio reale è 5, però potrebbe risultare interessante studiare altri valori modellando così altri veicoli come ad esempio il caso di uno scooter `N` = 2 o si può modellare il caso di un autobus con `N` = 35, 50, 60, ecc.

Organizzazione delle auto in *time buckets*

Tornando al nostro strumento, la prima operazione di cui si occupa *car-analysis.py* consiste nell'aprire il file *allvehicles.csv* del dataset considerato

e successivamente di memorizzare le informazioni contenute in ciascuna riga in una lista denominata `cars`. In realtà, ciascun elemento di questa lista è una classe, dal nome `CarInfo` appositamente implementata in un modulo chiamato *carmodule.py*, i cui attributi corrispondono a quelli illustrati al momento della scrittura di *allvehicles.csv* in *car-writer.py*, quindi:

- `car_id`: identificativo dell'auto
- `starttime`: tempo di partenza del veicolo
- `stoptime`: tempo di fine corsa del veicolo
- `long_start`: coordinata di longitudine di partenza
- `lat_start`: coordinata di latitudine di partenza
- `long_stop`: coordinata di longitudine di arrivo
- `lat_start`: coordinata di latitudine di arrivo

Dopo aver riempito la lista `cars`, si memorizza il numero di veicoli contenuti in `cars` in una variabile denominata `starting_cars` e si è scelto di organizzare le auto in piccoli "contenitori temporali" (`time_buckets`) in cui ciascun contenitore rappresenta una mezz'ora. Per fare ciò, si è riempito il dizionario `time_buckets` semplicemente andando a verificare in quale slot di mezz'ora ciascun veicolo rientrasse (ad esempio tra i secondi 0-1800 oppure 1800-3600, 3600-7200 e così via) in base al loro timestamp di `starttime` o `stoptime`, avendo come risultato un dizionario la cui chiave è rappresentata da una tupla di due valori indicanti i secondi della fascia di mezz'ora, mentre il valore corrisponde ad una lista di veicoli, cioè i veicoli appartenenti a quell'intervallo di tempo.

La funzione obiettivo

Il passo successivo consiste nel processo di *matching* vero e proprio tra i veicoli; al fine di garantire il più alto numero di *matching* possibile, il nostro

algoritmo si è basato su di una **funzione obiettivo** definita come segue:

$$u(k) = \alpha\frac{d_{kj}}{D} + \beta\frac{|t_k - t_j|}{T}, \quad \alpha, \beta \in [0,1], \quad \alpha + \beta = 1 \tag{2.1}$$

dove:

- d_{kj} rappresenta la distanza tra il veicolo k ed il veicolo j, calcolata secondo la (2.2)
- t_k rappresenta il timestamp del veicolo k
- t_j rappresenta il timestamp del veicolo j
- D è la distanza presa in input
- T è il tempo preso in input

$$d_{kj} = 2\arcsin(\sqrt{(\frac{\phi_2 - \phi_1}{2})^2 + \cos\phi_1 \cos\phi_2 \sin(\frac{\theta_2 - \theta_1}{2})^2})R \tag{2.2}$$

con:

- ϕ_1 è la latitudine della prima auto
- ϕ_2 è la latitudine della seconda auto
- θ_1 è la longitudine della prima auto
- θ_2 è la longitudine della seconda auto
- R è il raggio della Terra pari a 6367 km

L'applicazione della formula (2.1) fa sì che per il veicolo k la funzione obiettivo sia monotona crescente in maniera tale da garantire che il numero di *matching* possibili con il veicolo k cresca, anche se, come vedremo, i *matching* si baseranno su valori piccoli di $u(k)$ dato che il crescere di tale valore indica che due veicoli sono distanti a livello di caratteristiche.
I parametri α e β sono dei parametri particolari, essi devono essere compresi

tra 0 ed 1 e la loro somma deve essere comunque pari ad 1 per permettere la normalizzazione di $u(k)$ così che anch'esso possa rientrare nell'intervallo [0,1]. Il fatto di poterli variare permette di verificare il comportamento della riduzione delle auto nel caso si dia più peso al tempo piuttosto che alla distanza.

L'algoritmo di matching

Il procedimento di scansione della lista `cars` effettuato dall'algoritmo di *matching* implementato, richiede 3 cicli annidati del costrutto `for` in quanto,

1. il primo `for` serve per scandire tutti quegli insiemi di veicoli che abbiamo organizzato in precedenza in *time buckets*
2. il secondo ciclo serve per recuperare le informazioni di ogni auto da confrontare
3. il terzo `for` serve per recuperare le informazioni delle auto da confrontare con quelle del ciclo precedente per verificarne o meno la possibilità di *matching*

Nonostante questo procedimento possa sembrare aggravato a livello computazionale dal fatto di dover utilizzare tre cicli `for`, tuttavia il processo viene alleggerito dal fatto di dover scandire mini-gruppi di veicoli organizzati nei *time bucket*, così da evitare di dover confrontare tra di loro veicoli molto distanti sia in tempo che distanza in quanto con alta probabilità non *matcheranno* tra di loro, permettendo quindi di risparmiare del tempo inteso computazionalmente.
Durante questa fase di scansione oltre al recupero delle informazioni dei due veicoli da confrontare, vengono calcolate le distanze tra i punti iniziali delle due auto e le distanze tra i punti finali, secondo la (2.2) e i due veicoli *matcheranno* se:

1. ovviamente avranno identificativi diversi
2. la differenza tra i tempi iniziali delle due auto sia minore o uguale al tempo `T` accettato in input da *car-analysis.py*

3. la differenza tra i tempi finali delle due auto sia minore o uguale al tempo `T`

4. la distanza tra i punti iniziali delle due auto sia minore o uguale alla distanza `D` accettata in input da *car-analysis.py*

5. la distanza tra i punti finali delle due auto sia minore o uguale alla distanza `D`

Una volta che il confronto ha avuto successo, viene calcolato il valore $u(k)$ secondo la (2.1) e viene popolato un dizionario denominato `u_values_matrix` la cui chiave è l'identificativo del primo veicolo ed il valore è una lista di tuple, cioè delle coppie formate dall'identificativo del secondo veicolo ed il valore di $u(k)$. Nell'Algoritmo 1 si può vedere lo pseudocodice dell'algoritmo di *matching* che abbiamo sin qui descritto, da specificare che la funzione `calculateDistance()` si riferisce al calcolo tramite la (2.2), mentre `calculateUval()` fa riferimento al calcolo della funzione obiettivo (2.1).

Algorithm 1: Algoritmo di matching

```
Data: time_buckets := dictionary of vehicles managed by timestamps
begin
    for cars ∈ time_buckets.Values() do
        for first_car ∈ cars do
            id1 ⟵ first_car.getId()
            start_time1 ⟵ first_car.getStartTime()
            stop_time1 ⟵ first_car.getStopTime()
            long_start1 ⟵ first_car.getLongStart()
            lat_start1 ⟵ first_car.getLatStart()
            long_stop1 ⟵ first_car.getLongStop()
            lat_start1 ⟵ first_car.getLatStart()
            for second_car ∈ cars do
                id2 ⟵ second_car.getId()
                start_time2 ⟵ second_car.getStartTime()
                stop_time2 ⟵ second_car.getStopTime()
                long_start2 ⟵ second_car.getLongStart()
                lat_start2 ⟵ second_car.getLatStart()
                long_stop2 ⟵ second_car.getLongStop()
                lat_start2 ⟵ second_car.getLatStop()
                d_i ⟵
                 calculateDistance(lat_start1, lat_start2, long_start1, long_start2)
                d_f ⟵
                 calculateDistance(lat_stop1, lat_start2, long_stop1, long_stop2)
                if id1 ≠ id2 and |start_time1 − start_time2| ≤ T and
                 |stop_time1 − stop_time2| ≤ T and d_i ≤ D and
                 d_f ≤ D then
                    u ⟵ calculateUval()
                    t ⟵ (id2, u)
                    u_value_matrix.insert(id1, t)
```

Filtraggio della matrice degli $u(k)$

Il passo successivo compiuto da *car-analysis* consiste nel filtraggio della matrice composta dai valori $u(k)$ calcolati, ovvero di `u_value_matrix`, in quanto la matrice così come composta in questo momento, contiene in realtà solamente i possibili *matching* tra i veicoli ed inoltre si vuole evitare che delle stesse auto compaiano in diversi insiemi perchè risulterebbe non corretto il calcolo finale dei veicoli ridotti.
In definitiva, si vuole riempire una lista di liste che chiameremo `matched_cars` nel seguente modo:

1. per ogni elemento di `u_value_matrix` composto dalla coppia chiave-valore, viene riordinata la lista del sottoinsieme di veicoli che potrebbero essere *clusterizzati* con il veicolo indicato dalla chiave del dizionario, in ordine crescente in base al valore $u(k)$.

2. per ogni veicolo k, rappresentato dalla chiave del dizionario, si verifica che k non sia già stato inserito in un sottoinsieme di veicoli denominato `m_list` e se il test va a buon fine:

   (a) il veicolo k viene inserito in `m_list`

   (b) vengono considerati i primi `N` valori di $u(k)$ corrispondenti al secondo elemento della coppia formata dal veicolo j ed appunto $u(k)$

   (c) ciascun veicolo j viene inserito in `m_list` se e solo se: non è già stato inserito in precedenza in qualche altra `m_list` (questo controllo lo si effettua tenendo traccia di tutti i veicoli visitati fin'ora grazie ad un insieme chiamato `visited` costituito dagli identificativi dei veicoli che sono già stati inseriti nelle corrispondenti `m_list`) e l'`m_list` corrente non è "pieno", il che sostanzialmente significa che la sua dimensione è pari al parametro `N`

3. quando è stato riempito tutto l'`m_list` del veicolo k, questo viene aggiunto alla lista `matched_cars`

Quindi come risultato finale, l'algoritmo appena descritto, riempie una lista i cui singoli elementi sono a loro volta delle liste rappresentanti i sottoinsiemi

di veicoli che fanno *match* tra di loro.

Come si può ben intuire un algoritmo del genere mostra un comportamento *greedy* dato che per ogni *cluster* di veicoli, considera i primi `N` veicoli con valori di $u(k)$ migliori, intesi come pesi minori in quanto valori più piccoli indicano una certa affinità o similarità tra i veicoli in termini di vicinanza spaziale e/o temporale.

L'Algoritmo 2 mostra tutto il processo di filtraggio di `u_value_matrix` appena descritto.

Algorithm 2: Algoritmo di filtraggio della matrice degli $u(k)$

```
begin
    visited ⟵ ∅
    for <key, value> ∈ u_value_matrix do
        value.SortByUvalue()
        m_list ⟵ ∅
        if key ∉ visited then
            m_list.insert(key)
            visited.insert(key)
            for car_id ∈ value do
                if car_id ∉ visited then
                    if m_list.size()<N then
                        m_list.insert(car_id)
                        visited.insert(car_id)
                    else
                        break
```

Memorizzazione dei Risultati

Come passo finale, *car-analysis.py* si occupa di andare a calcolare il numero finale di auto che circolerebbero per la città analizzata, nel caso in cui si applicassero le funzionalità del *tool chain* da noi realizzato.
In particolare, la lista `matched_cars` viene ordinata in ordine decrescente in base alla grandezza di ciascun *cluster* di veicoli appartenente a tale lista. Successivamente viene effettuata una scansione di `matched_cars` ordinata e per ogni *cluster* di veicoli:

1. si somma la grandezza del *cluster* ad un contatore denominato `total_matched` e che quindi, alla fine del processo, conterrà il numero di auto totali che è stato possibile *matchare*

2. si considera ogni auto presente nel *cluster* e si scrive il suo identificativo in un file che vedremo fra un po'

Al termine della scansione di `matched_cars`, viene effettuata una sottrazione tra `starting_cars` e `matched_cars`, la quale sarà memorizzata in una variabile denominata `final_cars` e costituirà quindi il numero di veicoli restanti in circolazione per la città del dataset analizzato, valore che tornerà utile in fase di analisi.
Come passaggio finale, i dati rilevati saranno memorizzati in due file:

- *matched-cars-final-T-D-N-α-β.csv*: è il file scritto durante l'ultima scansione della lista `matched_cars`, dove T, presente nel nome del file, va sostituito con il limite `T` del tempo considerato in input dal modulo di analisi, D va sostituito col limite `D` della distanza, N con il numero di passeggeri `N` preso in input e quindi α e β saranno i valori di questi due parametri. Ogni riga del file è stata formattata separando tramite spazi vuoti la grandezza di ogni cluster dagli identificativi dei veicoli presenti nel *cluster*.
  E.g.:

```
5 c2178c3725f6fbb85129c9d2fec641f6 4011b4e36433035e9721ec1631fb4277 ...
5 49aed43e5be807ca502c42be9c6db526 83e568bf7cfaadc2a612e31fd3ee91cc ...
```

```
5 a4a0d96a08d9253235ee5c9bb0aab0e0 a0ec39b3bab8cd913a2c470536b77c76 ...

...

2 d9a3599186f80c6249325a2e5ddcd7dc 7c743baaef457cfee0737ca6ba7fb0a5
2 7c4109211766d014cb2d455f7cbf1536 526331be2d9d407015fae68f6cfa7035
2 8368d8f3678e05ccc8330b65fa7aaaa0 d7cef1086158629384d2dba6ca082128

...
```

- *car-pooling-α-β.dat*: dove α e β sono i pesi della formula (2.1), l'utilità di questo file (aperto in *append*) è data dal fatto che è formattato in maniera tale da poter essere preso in input da script di *gnuplot* per realizzare i grafici che vedremo nella sezione *Analisi e Risultati*. Ogni riga del file è costituita da attributi separati da spazi vuoti ed ogni spazio determina la separazione tra una colonna e l'altra proprio come se avessimo scritto una tabella; gli attributi memorizzati sono i seguenti:
  - il limite `T` del tempo considerato per l'analisi corrente
  - il limite `D` della distanza considerata per l'analisi corrente
  - il valore di α
  - il valore di β
  - il numero `N` massimo di passeggeri considerati per l'analisi corrente
  - il numero di auto ancora rimaste in circolazione (`final_cars`)
  - il numero di auto iniziali (`starting_cars`)

Quindi come si è ben intuito, in definitiva il processo compiuto da questo strumento consiste nella lettura di file *.csv* costituenti il dataset e dunque nella manipolazione e gestione delle tracce GPS, così da ottenere dei *cluster* di veicoli realizzati grazie ai limiti passati in input al *tool chain*.
Tale processo è illustrato con un esempio dalle seguenti figure.

Nelle Figure 2.1 e 2.2 si possono osservare le posizioni iniziali rispettivamente di 5 e 3 veicoli (rappresentati dai pallini colorati) all'interno della città di Bari con relativo tempo di partenza. Le informazioni relative all'esempio, sono state recuperate dal dataset che si riferisce a tale città con data 30 Aprile 2015 in cui, in questo specifico caso, si è eseguita un'analisi con distanza massima di 1 km e differenza temporale massima di mezz'ora (1800 s) tra i veicoli *matchati* e numero di passeggeri posto pari a 5.

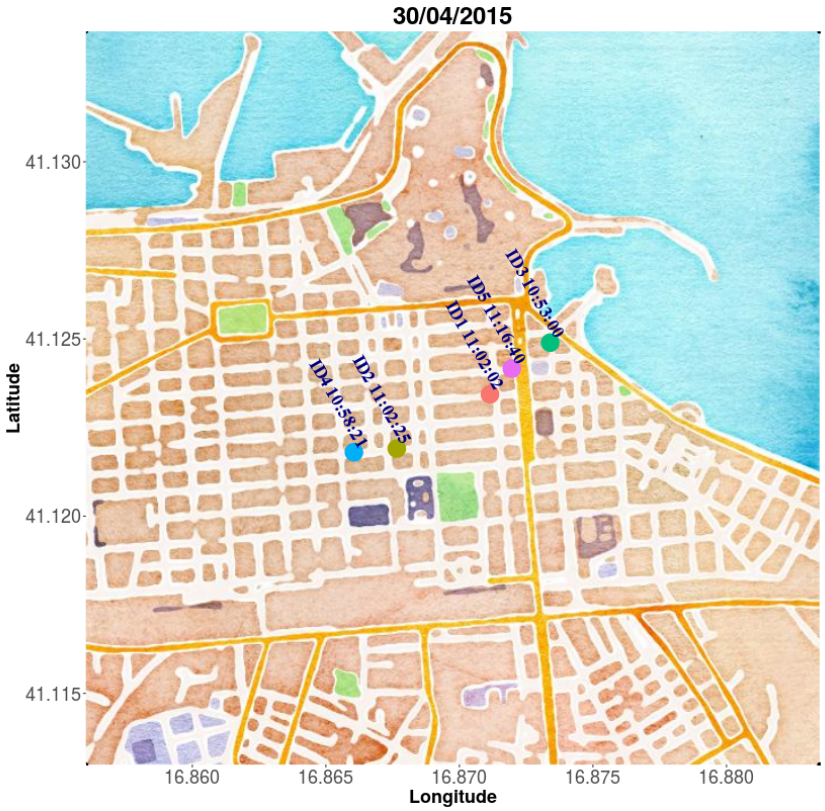

Figura 2.1: Esempio di 5 veicoli circolanti per la città di Bari
Nota: ciascun pallino colorato rappresenta un veicolo le cui label sono composte da una coppia in cui il primo elemento è l'identificativo del veicolo (per questioni di semplicità nella figura è reso generico dalla nomeclatura IDn), mentre il secondo elemento indica l'ora di partenza del veicolo.

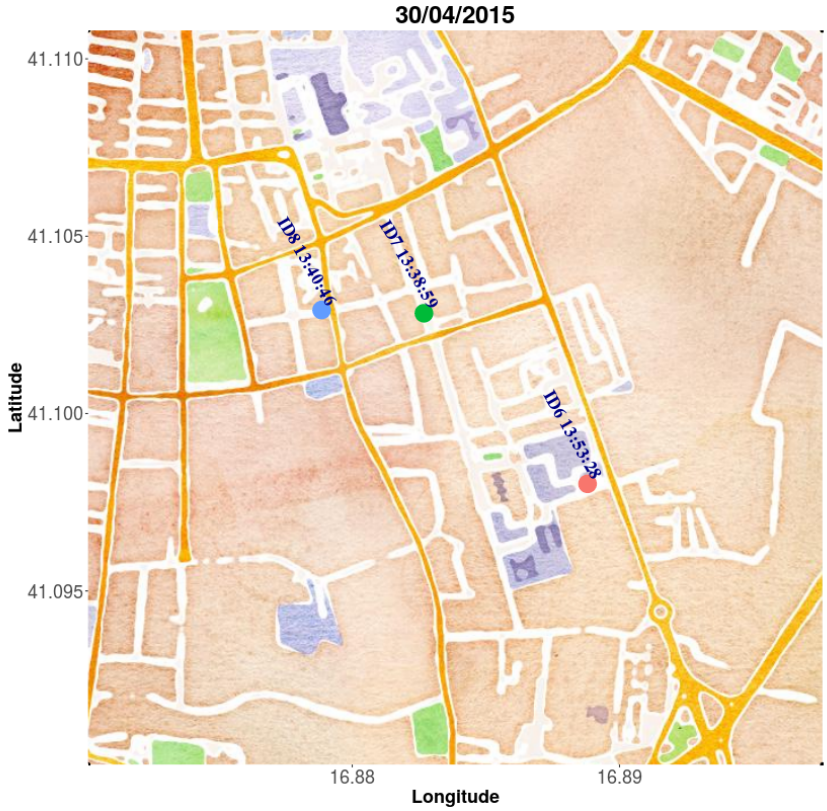

Figura 2.2: Esempio di 3 veicoli circolanti per la città di Bari

Nella successive figure (2.3 e 2.4), invece, si possono osservare le posizioni di arrivo dei veicoli sempre dello stesso dataset durante la stessa giornata, visivamente, anche grazie ai colori dei pallini, si può certamente notare che i veicoli si sono spostati e dalle label si osserva anche che i tempi sono maggiori rispetto alle figure precedenti.

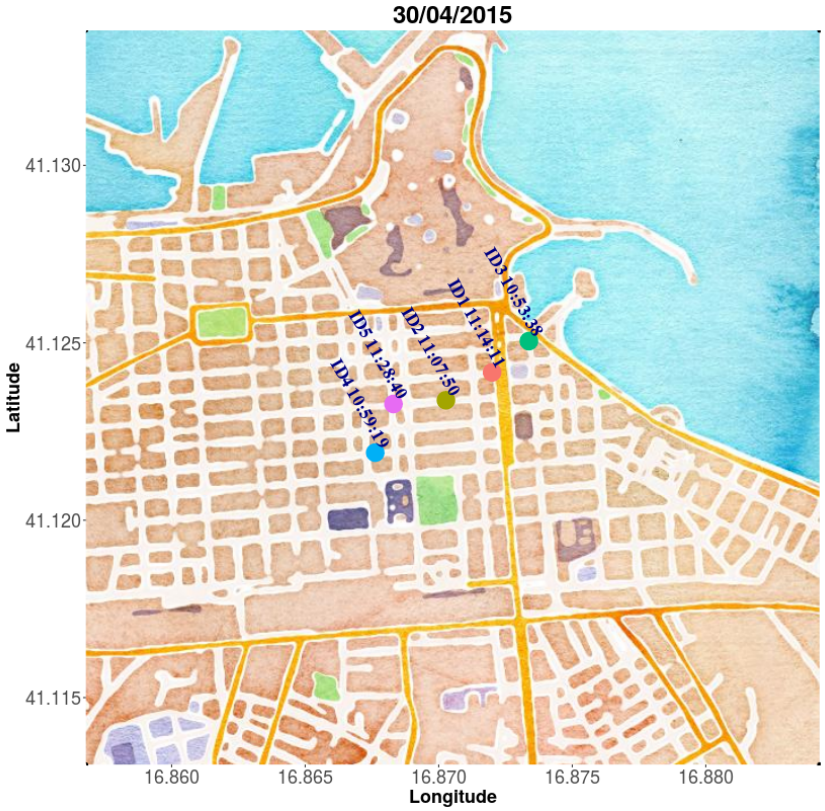

Figura 2.3: Posizioni finali dei 5 veicoli della Figura 2.1

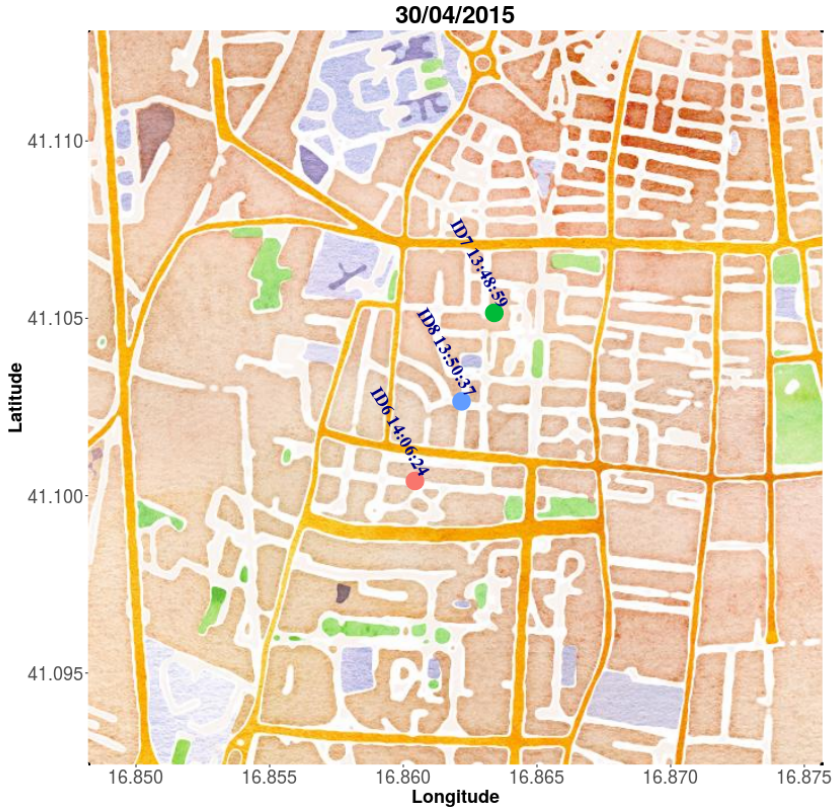

Figura 2.4: Posizioni finali dei 3 veicoli della Figura 2.2

Come risultato finale dell'analisi si sono ottenute due righe nel file *matched-cars-final-1800-1000-5-0.5-0.5.csv* rappresentanti i *cluster*:

```
5 c4a9eebc6a4a882aeb8dd3895a803c32 3f4ac0338041f9c6243f788f335ed2b1
2cb9ae4144fdf57069dc5a8ac0ddbcd6 e768a7df277f9240a4af2378796e63f4
f33b0a0e671bee8ac1ee33473ad05092
3 1582a4ee46f5284ca15638e325824f7b a57f71298269f1e2e9c59c5c3559c2fd
865d9de4f53646018cd6f664f5ebf82e
```

il quale può essere visivamente mostrato come nelle figure 2.5 e 2.6, dove il pallino risultante rappresenta, per l'appunto, il *cluster* di veicoli centrato nella media di coordinate di partenza e di arrivo rispettivamente dei 5 e 3 veicoli dell'esempio.

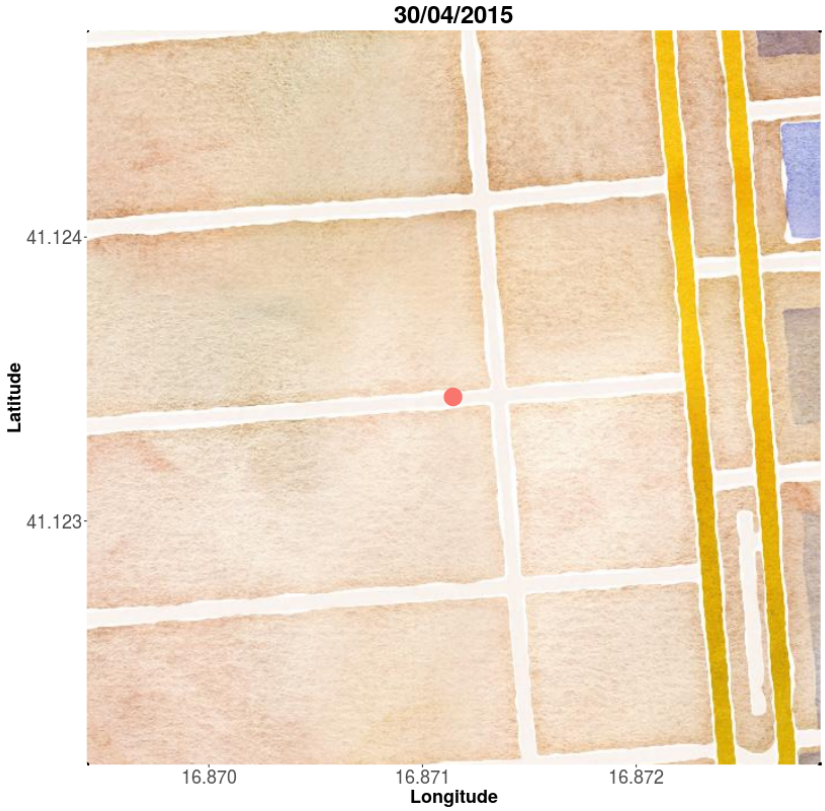

Figura 2.5: Cluster dei 5 veicoli *matchati* grazie a *car-analysis-py*

Figura 2.6: Cluster dei 3 veicoli *matchati* grazie a *car-analysis-py*

2.2.3 I Launcher

Come detto, il *tool chain* è stato reso il più generico possibile tanto è vero che vi è la possibilità di eseguire i due moduli: quello che si occupa della scrittura del file *allvehicles.csv* e quello che si occupa di eseguire un'analisi, in maniera indipendente l'uno dall'altro. Tuttavia, per effettuare un'analisi completa al variare dei parametri `T`, `D`, `N`, α e β, è risultato più utile implementare dei moduli, definiti *launcher* i quali permettono anche di lanciare più analisi; i *launcher* in questione sono:

- *launcher-car-files.py*: si occupa di richiamare il modulo *car-writer.py* ripetutamente su ogni singolo file del dataset che si vuole analizzare, all'interno del quale viene lanciato il comando:

```
python car-writer.py city_dir f
```

dove:

- *city_dir*: è il percorso della cartella del dataset da analizzare
- f: è il file che verrà aperto dal *car writer*

- *launch-analysis.py*: il quale si occupa semplicemente di eseguire il comando:

```
python car-analysis.py t d alpha beta N city_dir
```

combinando i valori di tempo, distanza, α, β ed N che sono contenuti in opportuni *array*.

- *launchEverything.sh*: prende in input il percorso del dataset da analizzare, rimuove un eventuale *allvehicles.csv* già presente dalla cartella di analisi ed esegue prima il *launch-car-files.py* e successivamente il *launch-analysis.py*.

Capitolo 3

Analisi e Risultati

Il tipo di studi che abbiamo voluto realizzare sono stati protesi a verificare se, dato un dataset iniziale di dati GPS di una qualsiasi città, è possibile aggregare tra di loro un numero di auto tali da permettere ai guidatori di condividere un'automobile e di diminuire dunque il numero di veicoli circolanti per la città considerata, con evidenti vantaggi anche per la viabilità all'interno della città stessa.
Quindi al variare dei parametri `T`, `D` ed `N` del modulo *car-analysis.py* verificare se è stato possibile realizzare una buona riduzione di veicoli o se magari è necessario introdurre una linea di autobus. Per il nostro tipo di studi abbiamo deciso di tenere i parametri α e β fissi al valore di 0.5. Tuttavia, abbiamo deciso di inserirli nella nostra funzione obiettivo, dati studi vigenti in letteratura per tematiche simili e quindi abbiamo deciso di utilizzarli per il nostro strumento realizzato in questo lavoro di tesi, in maniera tale da renderlo il più generico possibile e quindi fare in modo che possa essere utilizzato anche per studi futuri che permetterebbo di affacciarsi anche a diverse tematiche pur restando nel dominio della mobilità veicolare. Il fatto di poterli variare permetterebbe di effettuare studi di granularità diverse a livello spaziale e/o temporale, di fatti questi parametri si comportano come dei pesi nella funzione obiettivo e si potrebbe, ad esempio, analizzare il comportamento della riduzione dei veicoli dando più importanza al tempo o allo spazio. Un effetto collaterale che deriverebbe dal variare di α e β si manifesterebbe sicuramente

a livello dei vari sottoinsiemi (o *cluster*), ovvero cambierebbe l'associazione tra i veicoli facendo *matchare*, magari, due veicoli non associati in un'altra analisi impostata con valori diversi di questi parametri.

Nelle prossime sezioni saranno mostrati i risultati ottenuti dalle analisi da noi effettuate, riguardanti la riduzione di veicoli all'interno di una città, al fine di validare lo strumento realizzato e descritto in questo lavoro di tesi. Come campione si è scelto il dataset della città di Bari recuperando le tracce di mobilità in riferimento a 5 giovedì tra Marzo ed Aprile 2015 in quanto giorni feriali, per studiare in maniera verosimile il servizio di *car pooling* durante una giornata tipica di lavoro; i giorni in questione sono: 12 Marzo, 19 Marzo, 9 Aprile, 23 Aprile e 30 Aprile 2015.
Gli studi effettuati sono stati impostati in maniera tale da verificare l'impatto che può assumere il variare dei limiti in tempo, distanza e di passaggeri sulla riduzione generale dei veicoli all'interno dell'area urbana di Bari. Questi limiti (i cui valori si vedranno nelle prossime sezioni) sono stati scelti tenendo conto di esperienze di casi reali presenti in letteratura grazie a *survey* effettuate da altri ricercatori.
Inoltre è bene sottolineare che i risultati che vedremo sono sì frutto di uno studio che è voluto essere il più verosimile possibile, ma è stato condizionato dal numero dei veicoli presenti nei dataset giornalieri dato che il rilevamento delle tracce GPS è stato possibile solamente grazie al contributo dei veicoli che hanno fornito l'autorizzazione a partecipare al rilevamento delle proprie tracce di mobilità e che, inoltre, hanno sempre mantenuto attivo il dispositivo GPS durante il loro viaggio. Di fatti come abbiamo visto nel capitolo precedente, si sono dovuti scartare dall'analisi alcuni file i quali erano composti da poche righe rappresentanti sempre la stessa posizione, poichè il dispositivo GPS del veicolo interessato è stato tenuto acceso a veicolo fermo e dopo un po' è stato spento. In media i viaggi rilevati per dataset sono stati dell'ordine di 33000 a dispetto di circa 50000 file *.csv* di partenza presenti in ogni dataset ed è quindi chiaro che seppur essendo Bari una città non molto grande, questi dati non rappresentano la totalità delle auto che effettivamente ancora oggi viaggiano per la città.

3.1 Analisi al variare del tempo

In questa sezione vedremo i risultati degli studi effettuati analizzando la riduzione delle auto al variare del parametro T passato in input al modulo *car-analysis* in termini di secondi.
Da alcuni studi effettuati dalla comunità di ricerca è stato rilevato che in media, un guidatore che è disponibile a condividere la propria auto con altra gente, è disposto ad accettare ritardi oppure anticipi di 15 minuti sul suo solito orario quotidiano, purchè arrivi in orario nella propria sede di lavoro dopo aver accompagnato tutti i passeggeri e solamente in casi estremi è disposto ad accettare anche ritardi o anticipi di mezz'ora; è il caso di un guidatore che è disposto a rinunciare ad un po' di comodità pur di avere vantaggi a lungo termine sui costi totali che affronta mensilmente. Dunque abbiamo scelto di impostare la nostra analisi variando il tempo tra i 10 minuti e mezz'ora, aumentando via via il limite di 5 minuti. Abbiamo quindi impostato nel *launcher*, *launch-analysis* un *array* con valori pari a 600, 900, 1200, 1500 e 1800 secondi e lo abbiamo permutato con l'*array* delle distanze. Anticipiamo che per quanto riguarda le distanze i valori di analisi scelti sono pari a 250, 500, 750 e 1000 metri in quanto secondo alcuni studi realizzati tramite *survey*, è stato evidenziato che, in media, i guidatori sono disposti ad accettare deviazioni di 500-600 metri rispetto al loro percorso abitudinario, tollerando fino ad un limite massimo di 1 chilometro.

I seguenti grafici rappresentano analisi relative ai singoli giorni e, successivamente è stato effettuato un confronto tra i vari giorni sempre al variare del tempo.

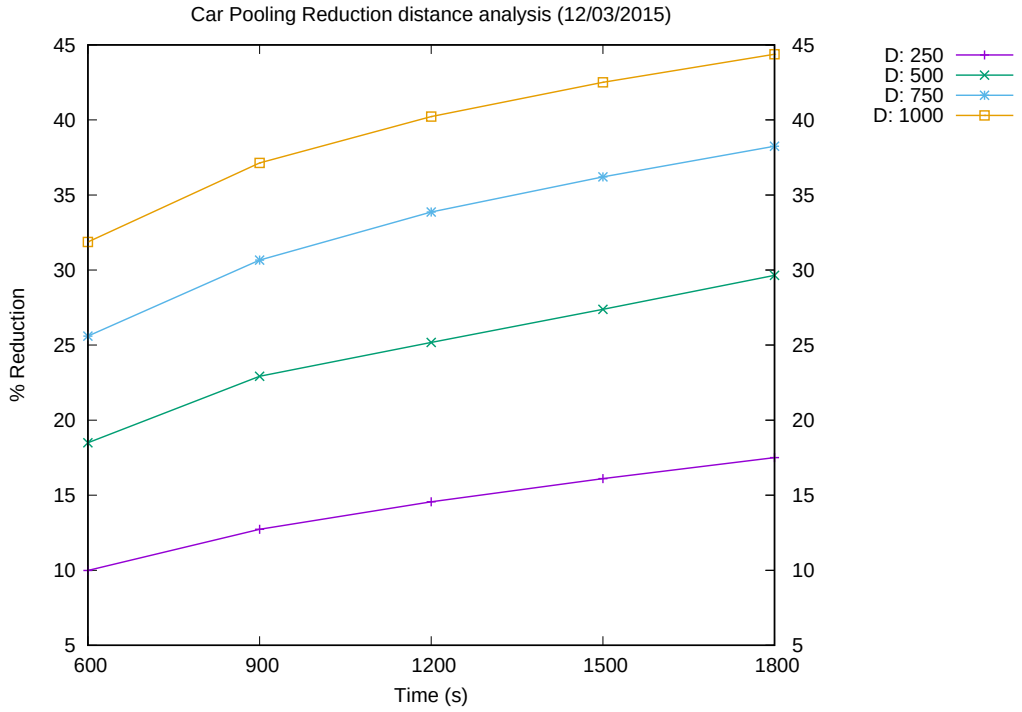

Figura 3.1: Percentuale di riduzione al variare del tempo per il giorno 12/03/2015

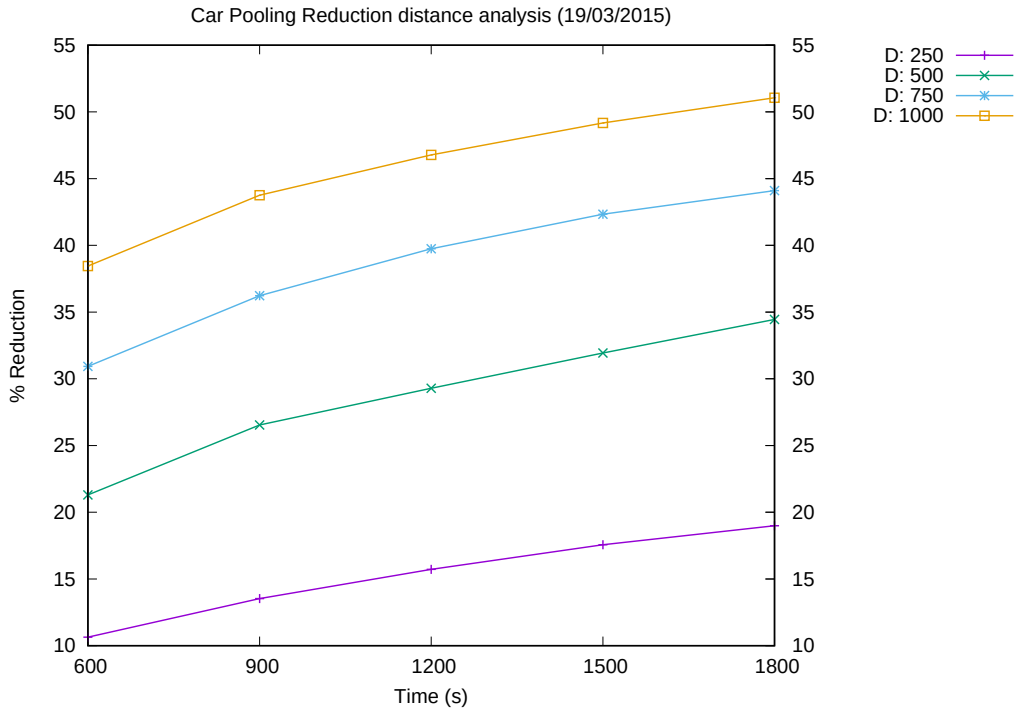

Figura 3.2: Percentuale di riduzione al variare del tempo per il giorno 19/03/2015

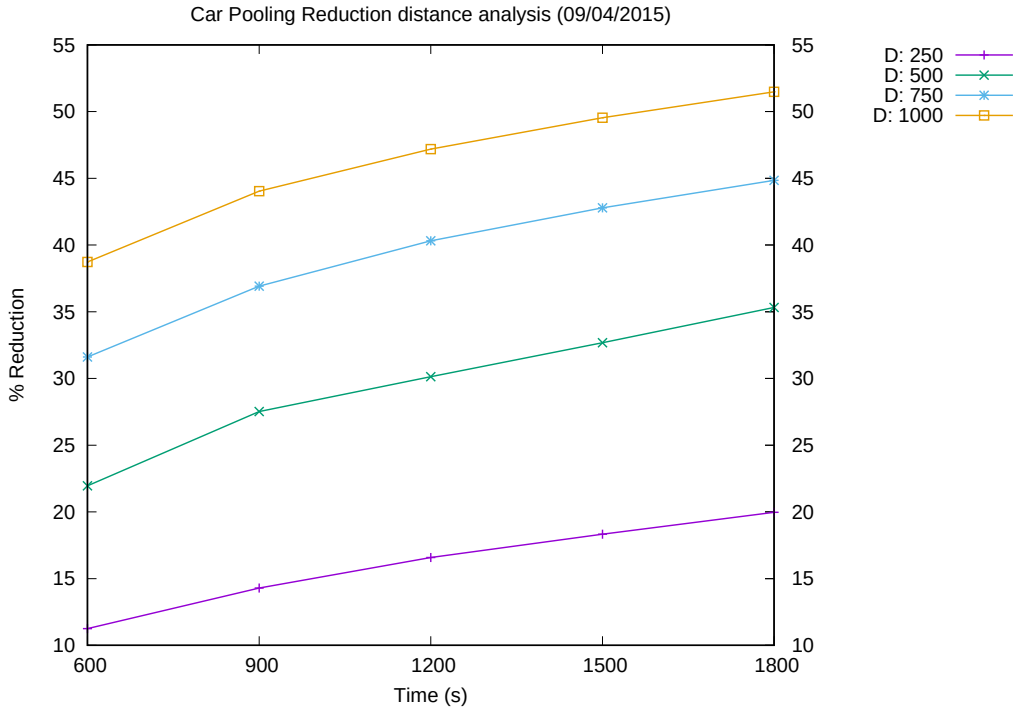

Figura 3.3: Percentuale di riduzione al variare del tempo per il giorno 09/04/2015

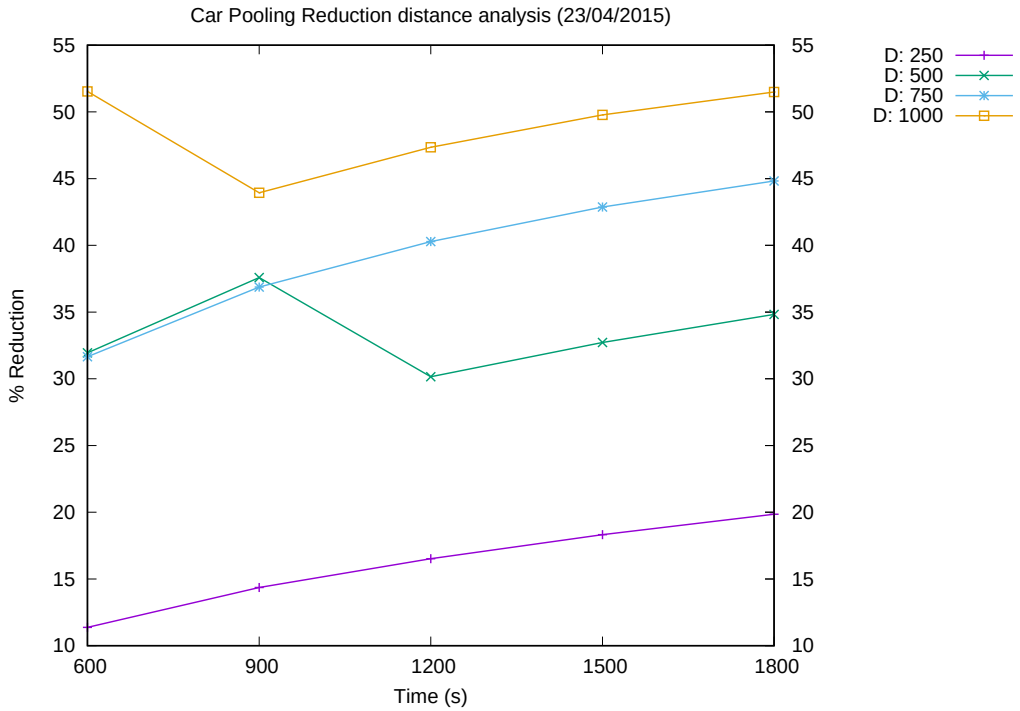

Figura 3.4: Percentuale di riduzione al variare del tempo per il giorno 23/04/2015

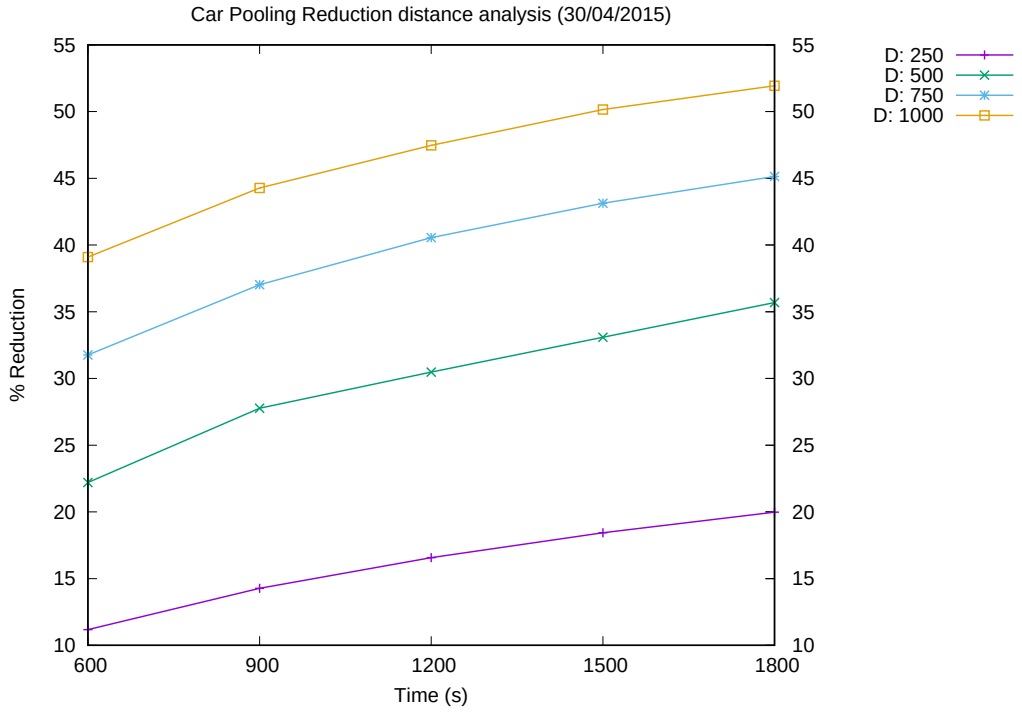

Figura 3.5: Percentuale di riduzione al variare del tempo per il giorno 30/04/2015

Come si può ben notare, le Figure dalla 3.1 alla 3.5 mettono in evidenza l'andamento della percentuale di riduzione delle auto fissando la distanza definita in funzione del tempo. Infatti per ogni giorno, ciascuna curva dei grafici rappresenta una particolare distanza (si veda la legenda) ed ogni punto tracciato sulla curva (sia esso un asterisco, una croce, un quadrato od una x) indica la combinazione con i tempi 600, 900, 1200, 1500, 1800 s.
I risultati ottenuti evidenziano che vi è una crescita lineare della percentuale di riduzione delle auto circolanti per la città di Bari, in maniera monotona crescente. Molto importante è osservare la curva corrispondente alla distanza dei 1000 metri perchè in questo caso si è ottenuta una percentuale di riduzione superiore al 50% (a parte per il giorno 12 Marzo 2015 in cui la percentuale massima si attesta intorno al 45%), il che in un caso reale avrebbe un impatto significativo importante sui benifici della città stessa. Solamente il giorno 23

Aprile 2015 presenta una "lieve anomalia" in quanto le curve corrispondenti alle distanze 500 e 1000 m presentano un decremento rispettivamente tra i secondi 900-1200 e 600-900, anche se in realtà, dopo questo lieve decremento, le curve tornano a crescere costantemente con i valori massimi che in fine si attestano in percentuali simili a quelle degli altri giorni, ovvero intorno al 35% per quanto riguarda la curva dei 500 m e 52% per la curva dei 1000 m. Questo è un comportamento logicamente comprensibile e che modella verosimilmente ciò che ci si aspetterebbe in un caso reale in quanto, se un guidatore diventa più permessivo sui tempi di attesa di un'altra persona, a distanze più elevate potrebbe accompagnare sempre più gente a lavoro o a casa.

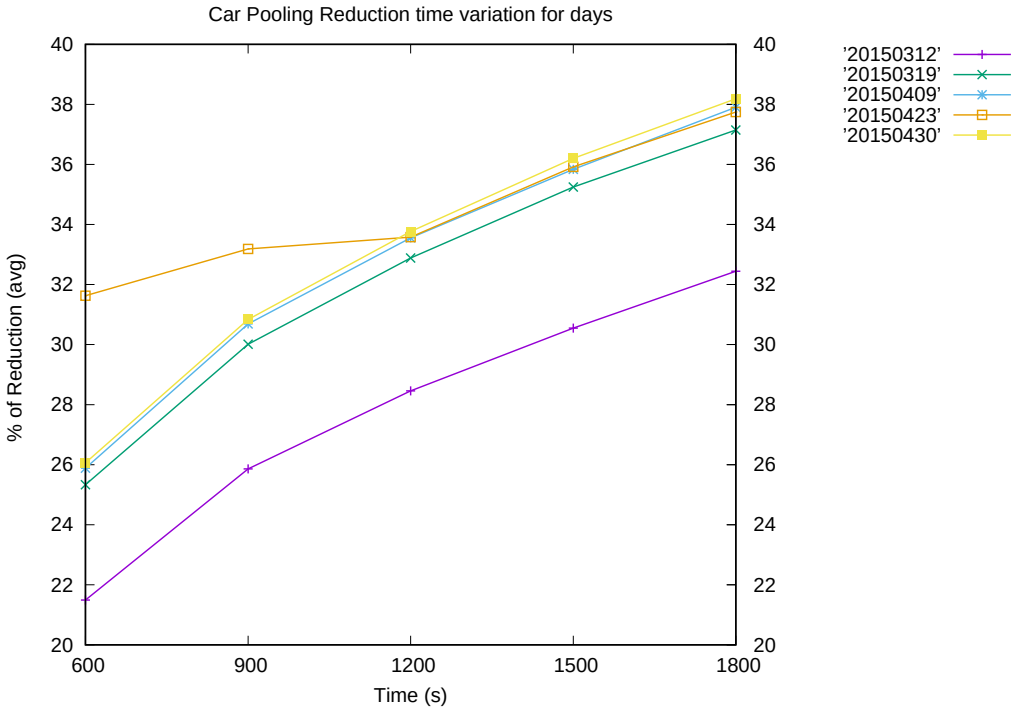

Figura 3.6: Percentuale media di riduzione al variare del tempo. Confronto tra giorni

Nella figura 3.6 viene effettuato un confronto tra i vari giorni, ciascuno corrispondente ad una curva del grafico, dove è stato necessario calcolare

la percentuale media di riduzione fissando la distanza e variando il tempo rappresentato sull'asse delle ascisse nel grafico in figura. Da questo confronto si ricava che nell'arco della giornata, il *matching* effettuato dall'algoritmo del *car-analysis.py* ha ottenuto una percentuale media di riduzione con valori che si attestano tra il 33 ed il 38% che tuttavia seppur non altissima, rappresenta una buona media.

3.2 Analisi al variare della distanza

In questa sezione visualizzeremo i risultati delle analisi effettuate al variare della distanza. Come già anticipato, si è impostato l'*array*, il cui singolo elemento è il parametro `D` del *car-analysis.py* con i valori di 250, 500, 750, 1000 metri.
Dai grafici delle Figure da 3.7 a 3.11 si nota che anche in questo caso, si presenta un comportamento delle curve motono crescente, però in questo caso ogni curva corrisponde ad un tempo `T` rientrante nell'intervallo 600-1800 s, messo in relazione con i valori della distanza sull'asse delle ascisse.
Osservando l'analisi da questa prospettiva, si osserva che la percentuale di riduzione si attesta sempre intorno al 51% con sempre "l'anomalia" del giorno 23 Aprile 2015 dove il decremento, infatti, si ha per quanto riguarda le cruve col tempo pari a 600 e 900s e successivo incremento fino al 51% circa per `T` = 600s, cosìcome, il giorno 12 Marzo 2015 presenta una percentuale di riduzione che non va oltre il 44% nella curva dei 1800 s. Dati entrambi questi studi effettuati in tempo e spazio confrontando, i grafici tra di loro si può ricavare che la distanza influirebbe maggiormente sulla possibilità di aggregare più veicoli tra di loro, rispetto al tempo.
In particolare questa ipotesi è rafforzata dal confronto realizzato tra i vari giorni in cui si è tenuto fissato il tempo `T` e si è calcolata la percentuale media di riduzione in funzione della distanza (Figura 3.12). Di fatti si nota subito che in questo caso, la percentuale media di riduzione dei veicoli si avvicina ad un buon valore, pari a circa il 49%.

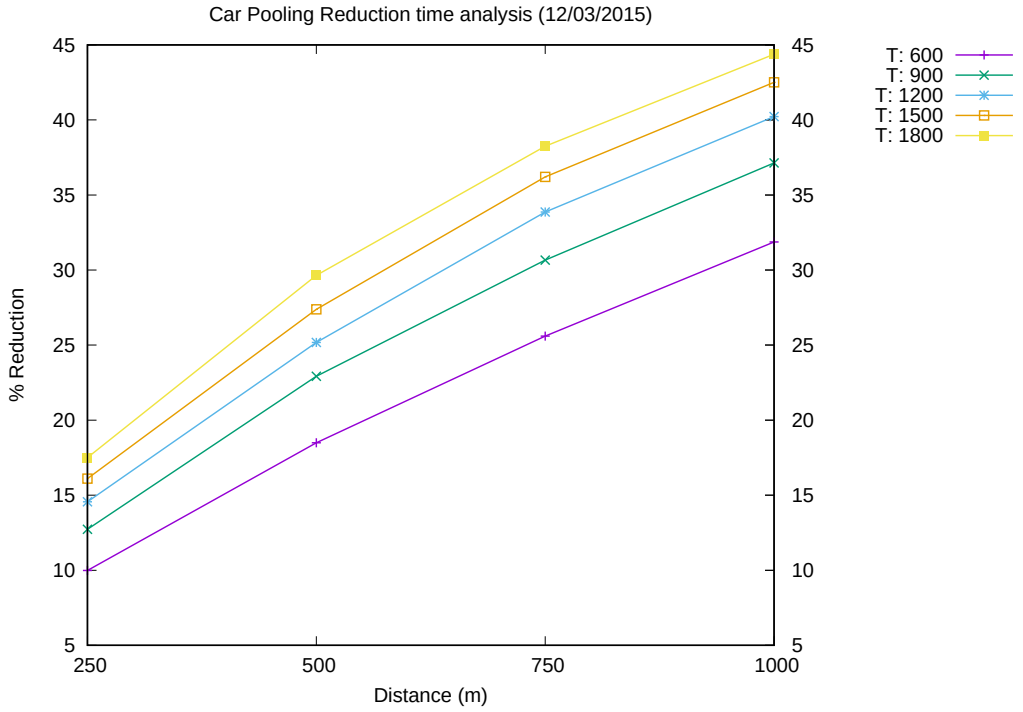

Figura 3.7: Percentuale di riduzione al variare della distanza per il giorno 12/03/2015

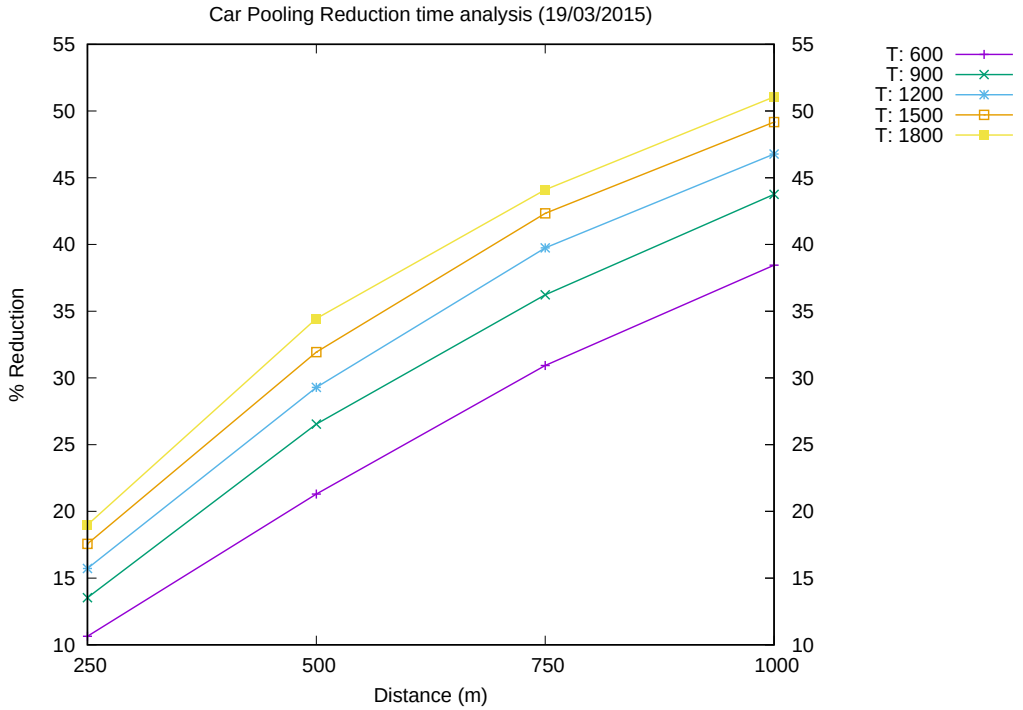

Figura 3.8: Percentuale di riduzione al variare della distanza per il giorno 19/03/2015

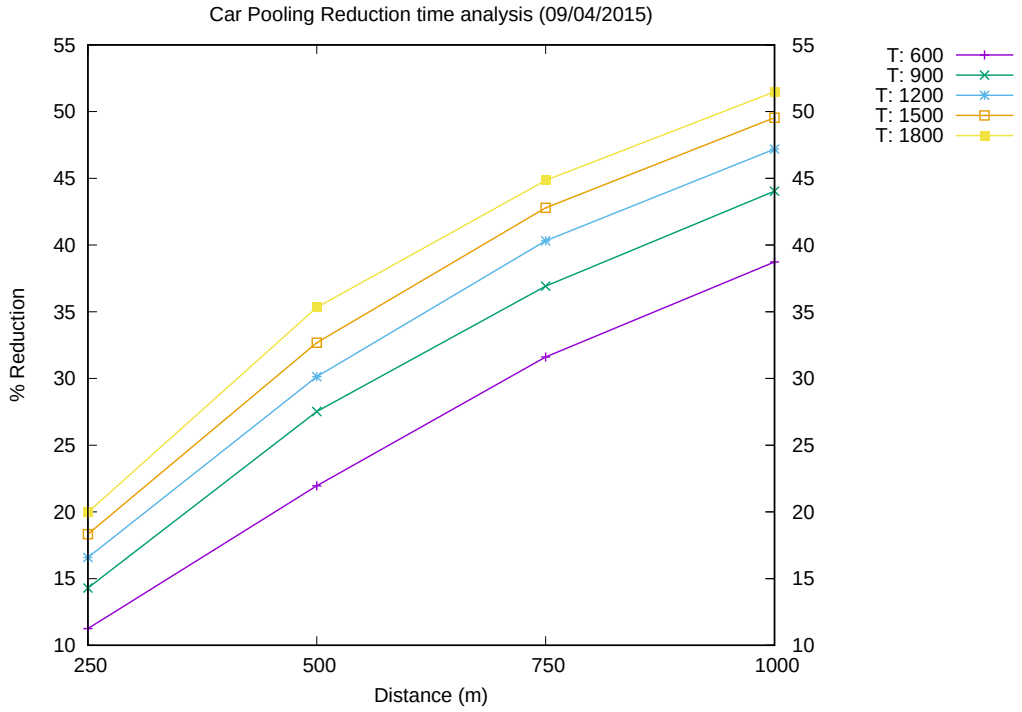

Figura 3.9: Percentuale di riduzione al variare della distanza per il giorno 09/04/2015

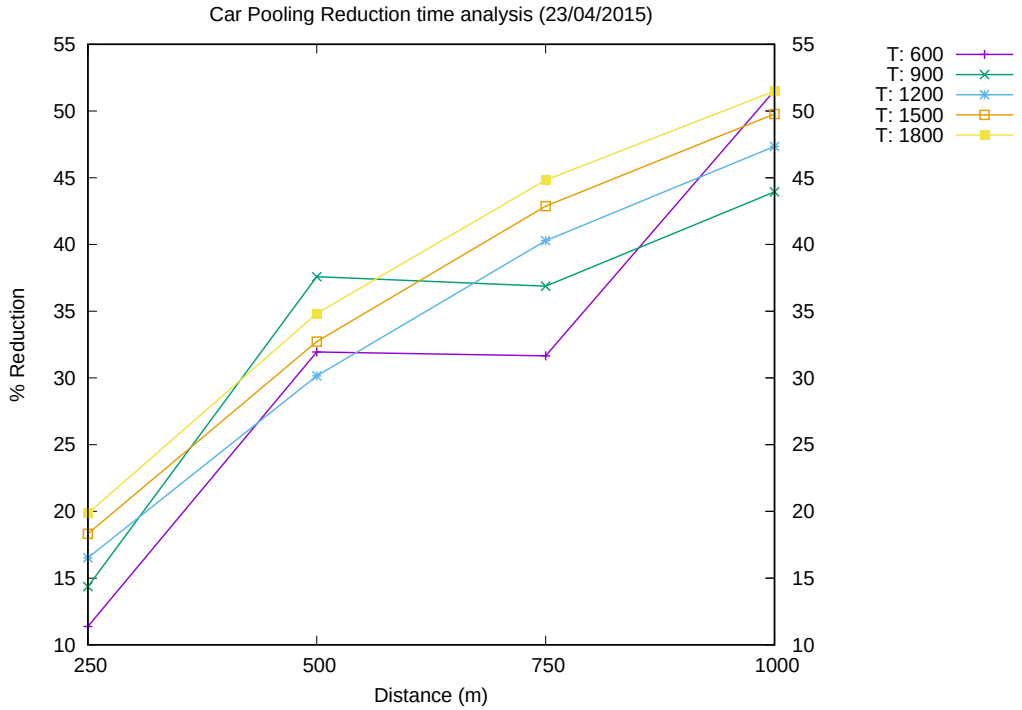

Figura 3.10: Percentuale di riduzione al variare della distanza per il giorno 23/04/2015

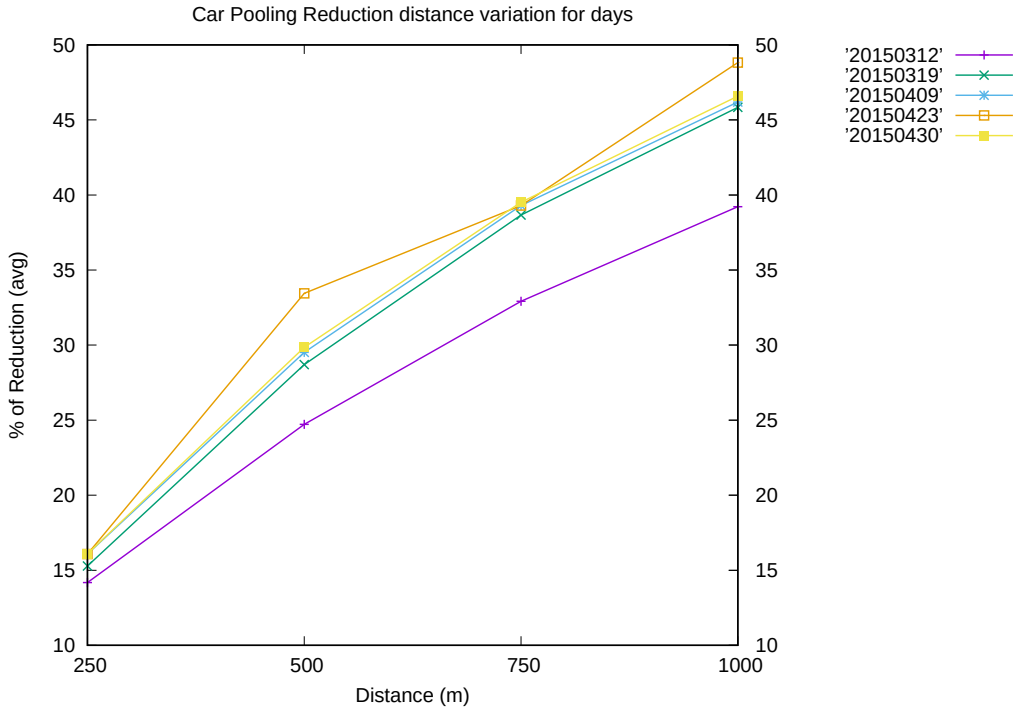

Figura 3.12: Percentuale media di riduzione al variare della distanza. Confronto tra giorni

3.3 Analisi sul numero di passeggeri medio

Uno studio interessante è stato effettuato andando ad analizzare il numero medio di passaggeri durante l'arco della giornata fissando un `N` massimo pari a 5, perchè ovviamente, è la capienza massima che un'auto può supportare e si è verificato quindi di aver ottenuto un numero di passeggeri tali da riempire in media le auto coinvolte nell'analisi, in quanto più auto si riescono a riempire con l'algoritmo di *matching* da noi implementato, meno auto ci saranno in circolazione nella città. I grafici che vanno dalla Figura 3.13 alla Figura 3.24 sono organizzati in questo modo:

- i primi 5 grafici mostrano il numero medio di passeggeri per auto tenendo fissa la distanza e considerando il variare del tempo durante le

5 giornate del dataset

- la figura 3.18 mostra un confronto tra i 5 giorni sempre considerando il variare del tempo

- i successivi 5 grafici mostrano il numero medio di passeggeri tenendo fisso il tempo e variando la distanza

- la figura 3.24 mostra un confronto tra i 5 giorni al variare della distanza

È interessante notare che quasi tutti i grafici convergono verso un unico valore medio che è quello di 3 passeggeri per auto; ciò evidenzia che vi potrebbero essere in media ancora altri e due posti liberi per ciascuna auto e questo è un risultato importante in quanto bisogna considerare il fatto che il dataset a nostra disposizione non contiene tutte le tracce di mobilità di tutti i veicoli di Bari. Infatti se disponessimo della totalità delle tracce GPS di tutti i veicoli, uno studio del genere fornirebbe dei risultati quantitativamente e qualitativamente migliori, pertanto consideriamo abbastanza buono quanto ottenuto in questo studio.

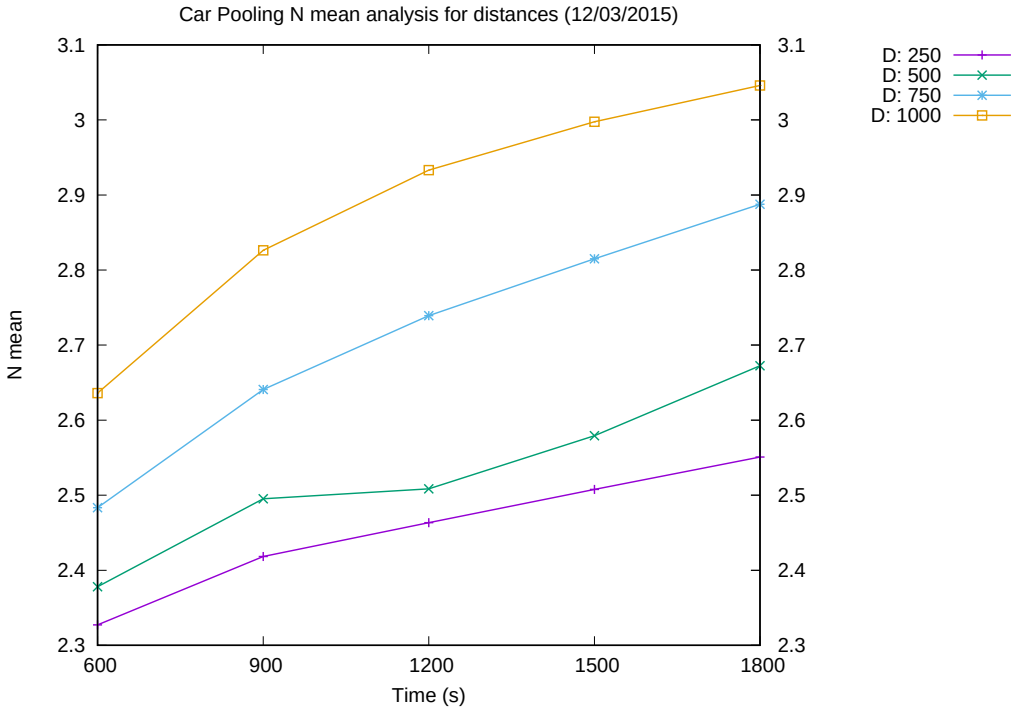

Figura 3.13: Numero di passeggeri medio al variare del tempo durante il giorno 12/03/2015

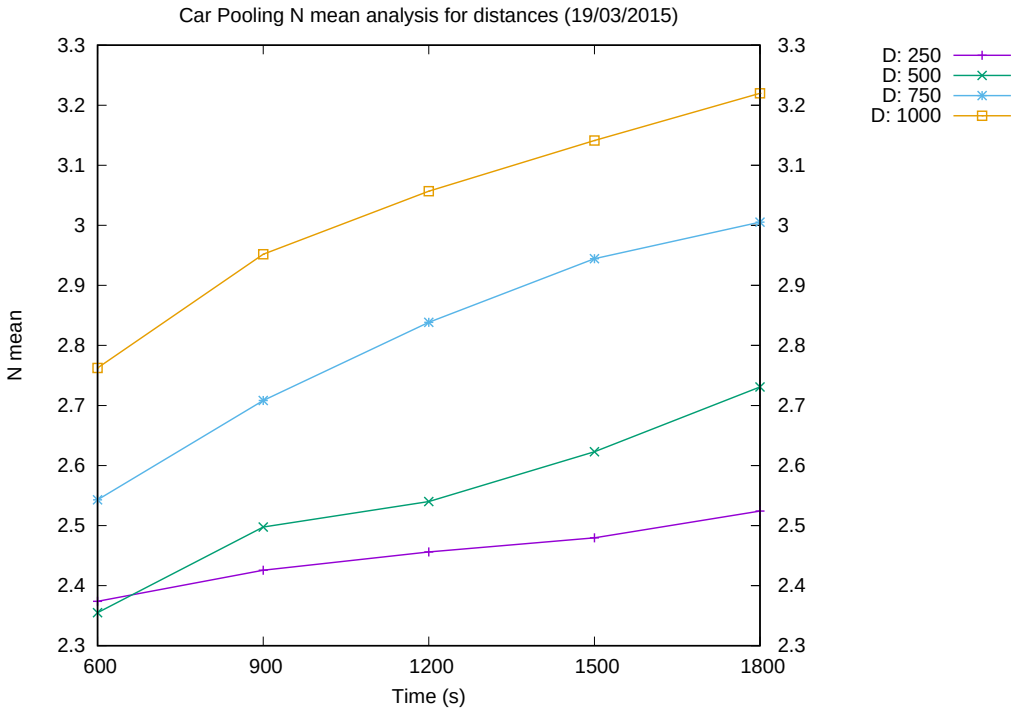

Figura 3.14: Numero di passeggeri medio al variare del tempo durante il giorno 19/03/2015

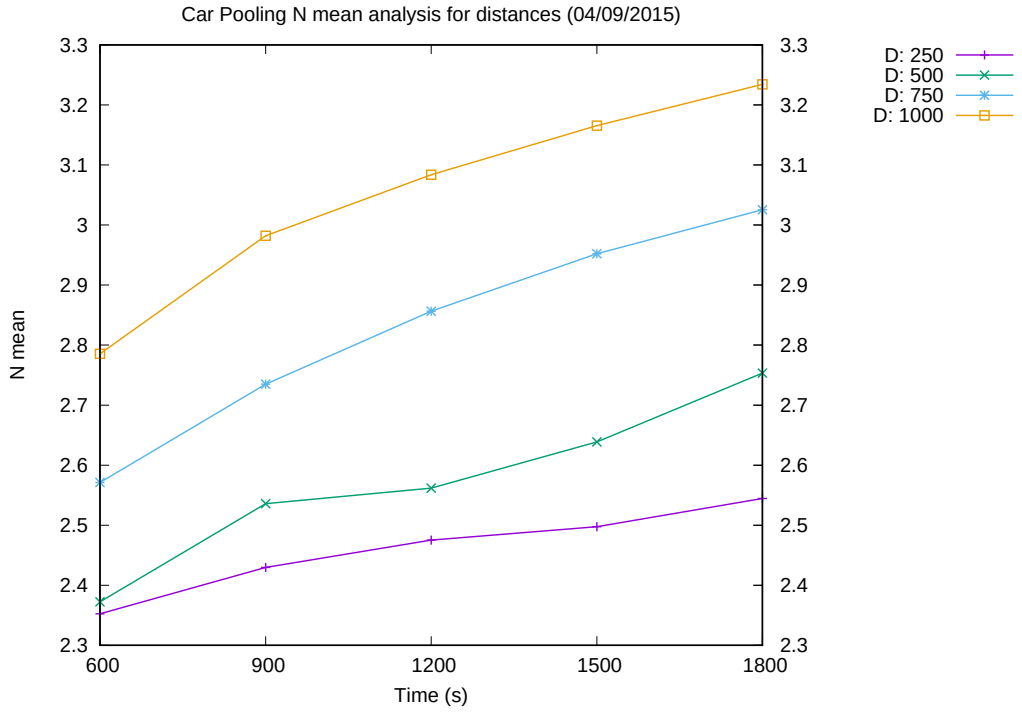

Figura 3.15: Numero di passeggeri medio al variare del tempo durante il giorno 09/04/2015

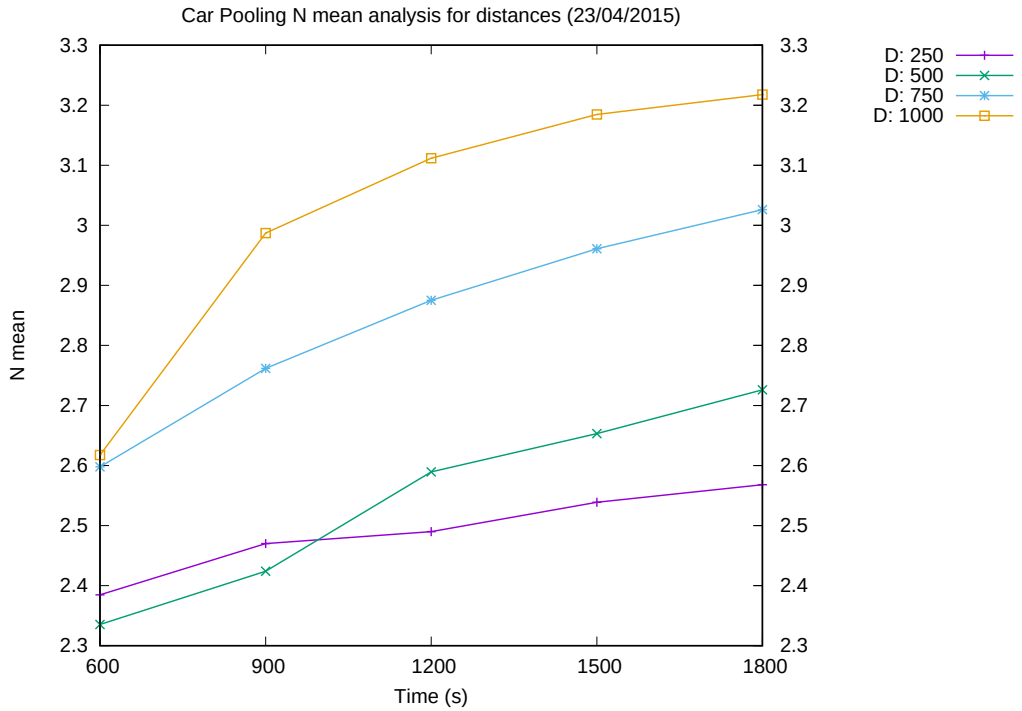

Figura 3.16: Numero di passeggeri medio al variare del tempo durante il giorno 23/04/2015

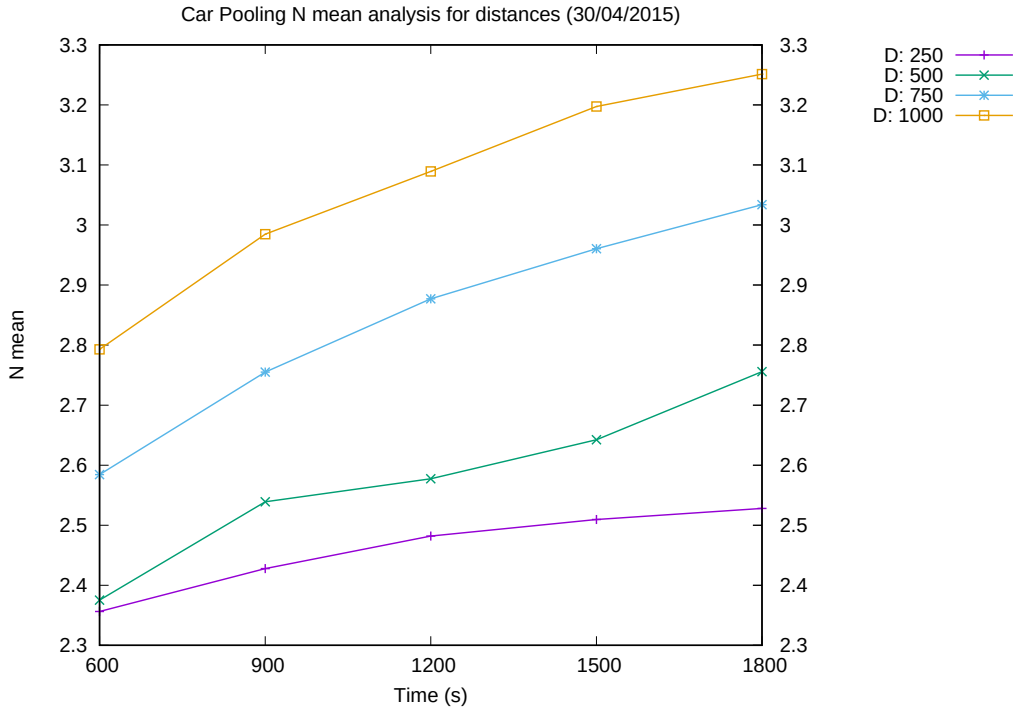

Figura 3.17: Numero di passeggeri medio al variare del tempo durante il giorno 30/04/2015

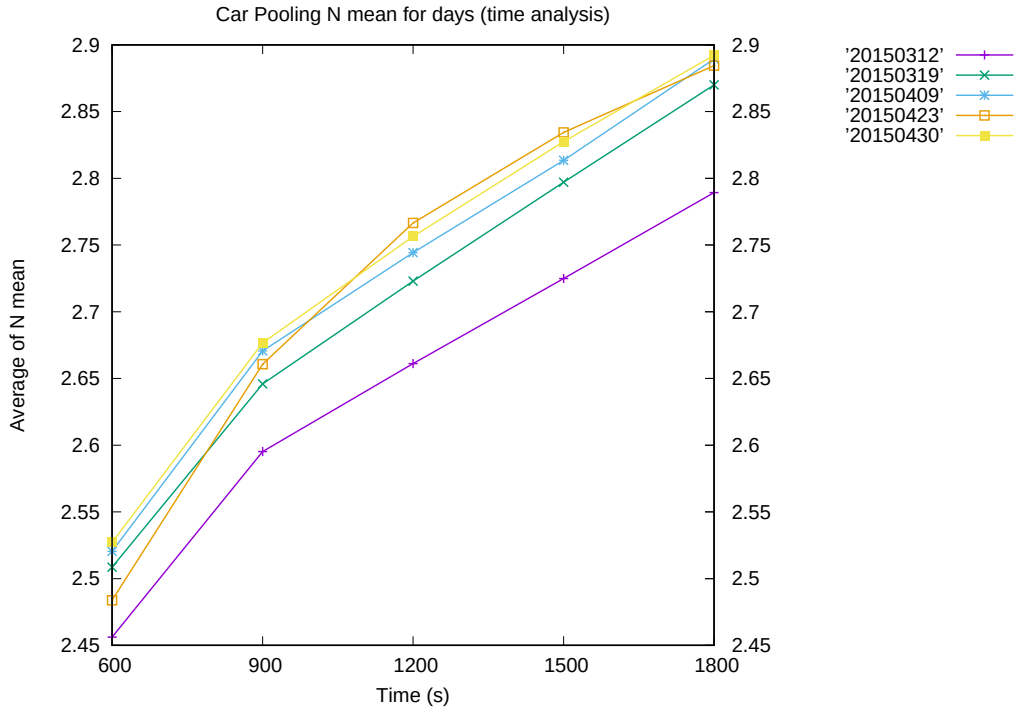

Figura 3.18: Numero di passeggeri medio al variare del tempo. Confronto tra giorni

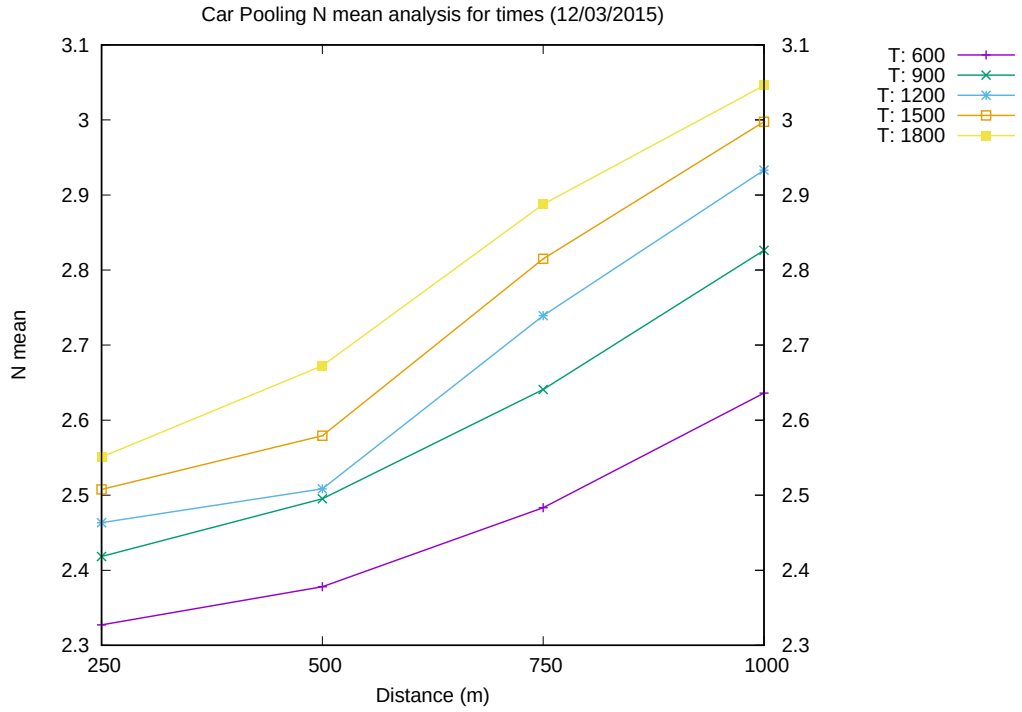

Figura 3.19: Numero di passeggeri medio al variare della distanza durante il giorno 12/03/2015

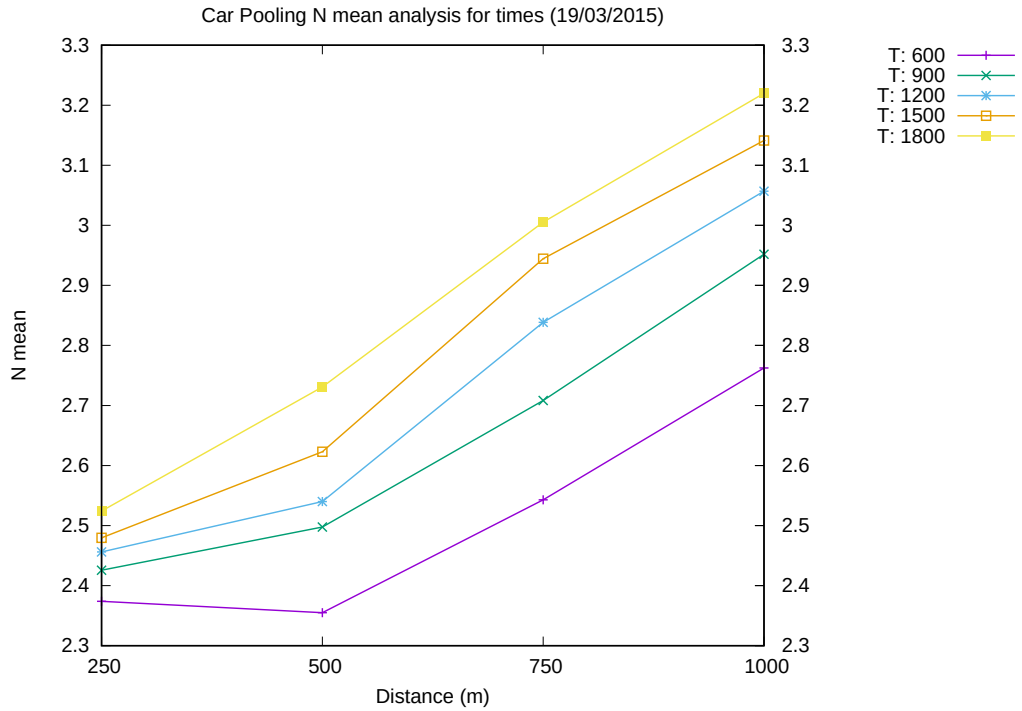

Figura 3.20: Numero di passeggeri medio al variare della distanza durante il giorno 19/03/2015

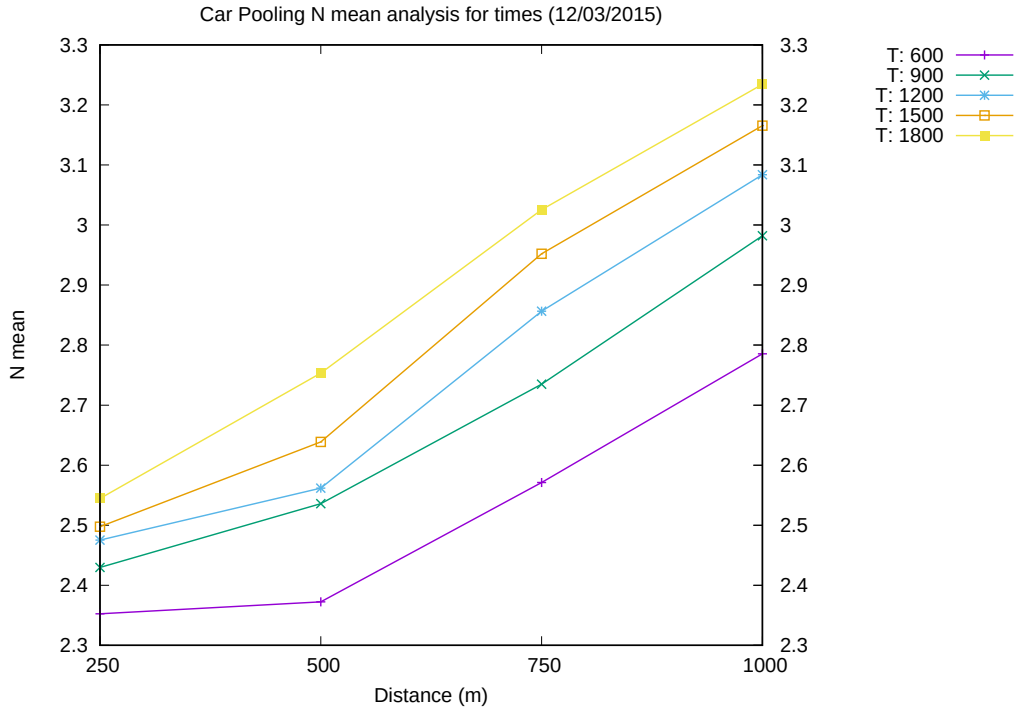

Figura 3.21: Numero di passeggeri medio al variare della distanza durante il giorno 09/04/2015

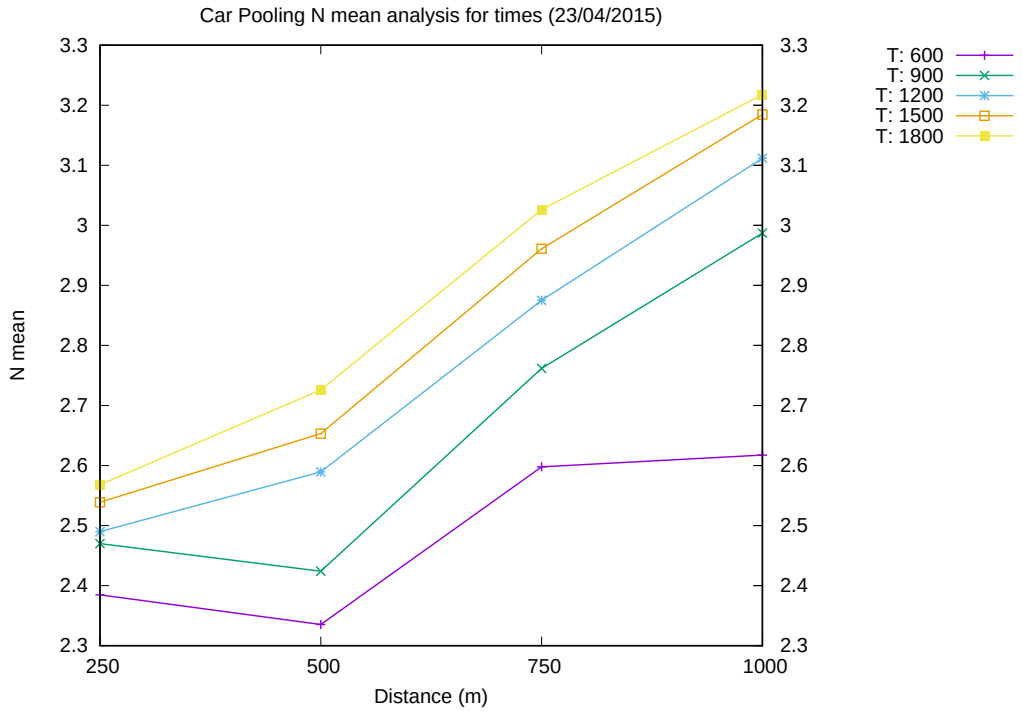

Figura 3.22: Numero di passeggeri medio al variare della distanza durante il giorno 23/04/2015

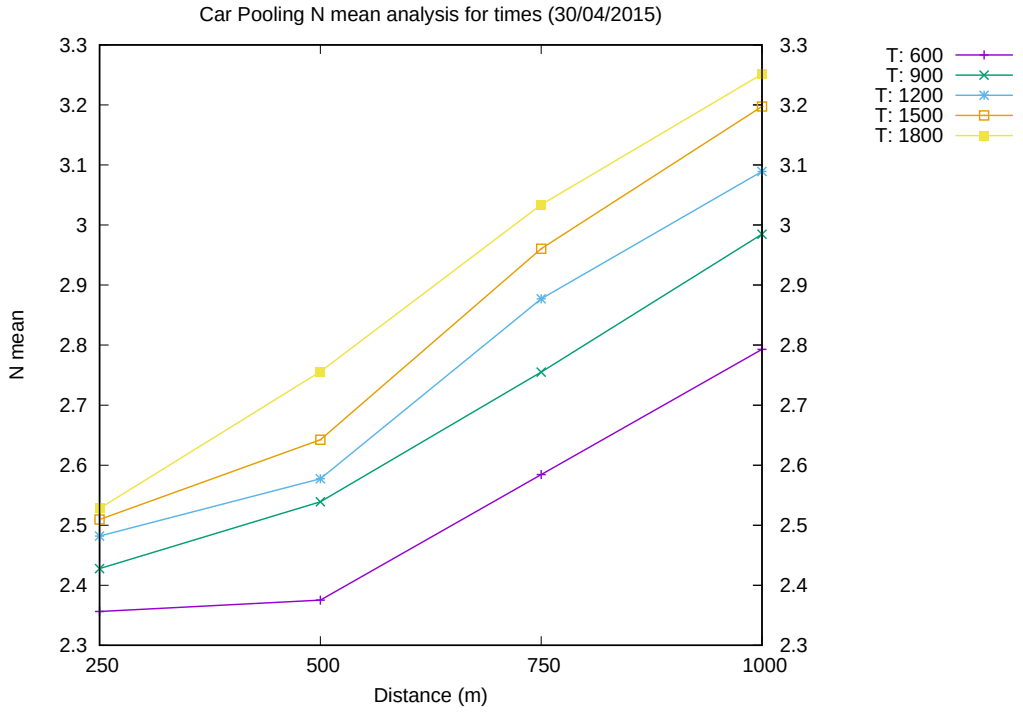

Figura 3.23: Numero di passeggeri medio al variare della distanza durante il giorno 30/04/2015

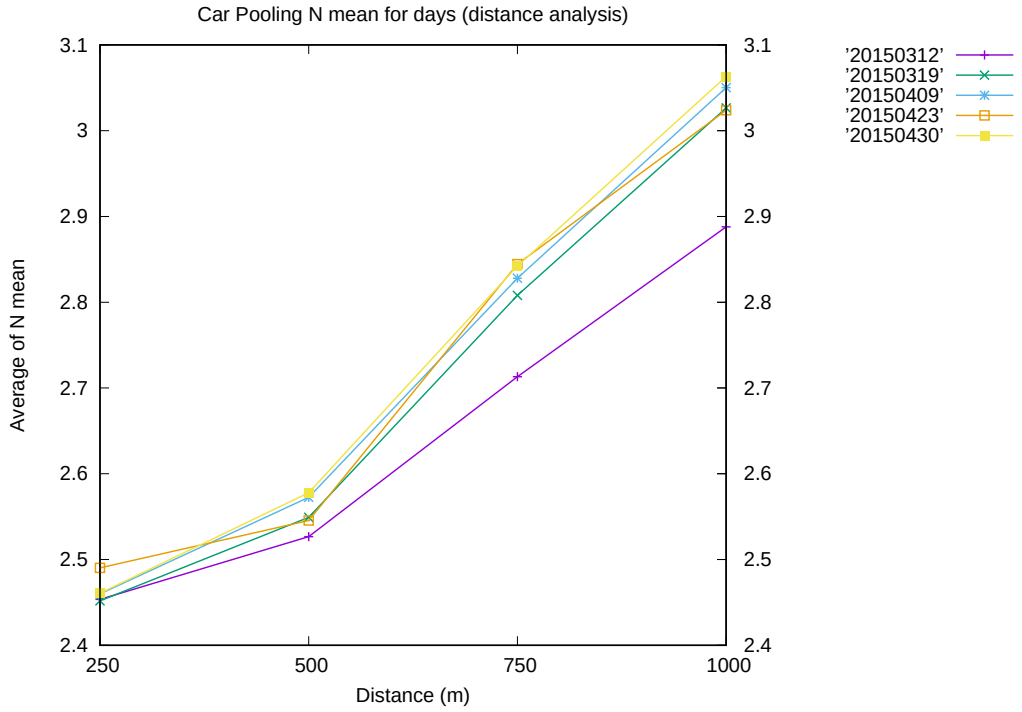

Figura 3.24: Numero di passeggeri medio al variare della distanza. Confronto tra giorni

3.4 Analisi al variare del numero di passeggeri

Un ulteriore studio che si è provato a realizzare, è andato ad interessare il parametro `N` del modulo `car-analysis.py`.
In particolare si è pensato di provare a variare il numero di passeggeri per veicolo, impostando un valore maggiore dei tipici 5 passeggeri per auto in quanto uno studio del genere potrebbe fornire degli spunti interessanti per la città e suggerire quindi al comune di inserire, magari, nuove linee di autobus per migliorare la qualità dei servizi e della viabilità offerti dal comune stesso.
Per questo tipo di analisi si è scelto di impostare i valori di tempo e distanza ai limiti massimi (quindi 1800 s e 1000 m) al fine di effettuare un *matching*

più alto possibile così che potesse essere più agevole verificare l'andamento della percentuale di riduzione, considerando anche il fatto che gli autobus in generale non sono collegati a nessun tipo di vincolo temporale e/o spaziale, mentre si è variato N tra i valori: 1, 2, 3, 4, 5, 10, 15, 20, 25, 30; ovviamente `N` = 1 non avrebbe alcun tipo di influenza ai fini dell'analisi in quanto presenta un *matching* pari allo 0%, ma comunque lo si è voluto inserire tra i valori da passare in input al `car-analysis.py` per presentare uno studio il più completo possibile nella sua interezza. Dal grafico in figura 3.25 si nota sorprendentemente che, per ogni giorno del dataset analizzato, la percentuale di riduzione presenta un incremento quasi esponenziale fino ad `N` pari a 10 per poi stabilizzarsi a valori che oscillano tra il 49 ed il 57-58%. Il fatto che da un certo numero di passeggeri in poi, non si riesce a migliorare la riduzione di veicoli suggerirebbe che in una città non molto grande come Bari, non vi sia la necessità di inserire ulteriori linee di autobus oltre a quelle già presenti.

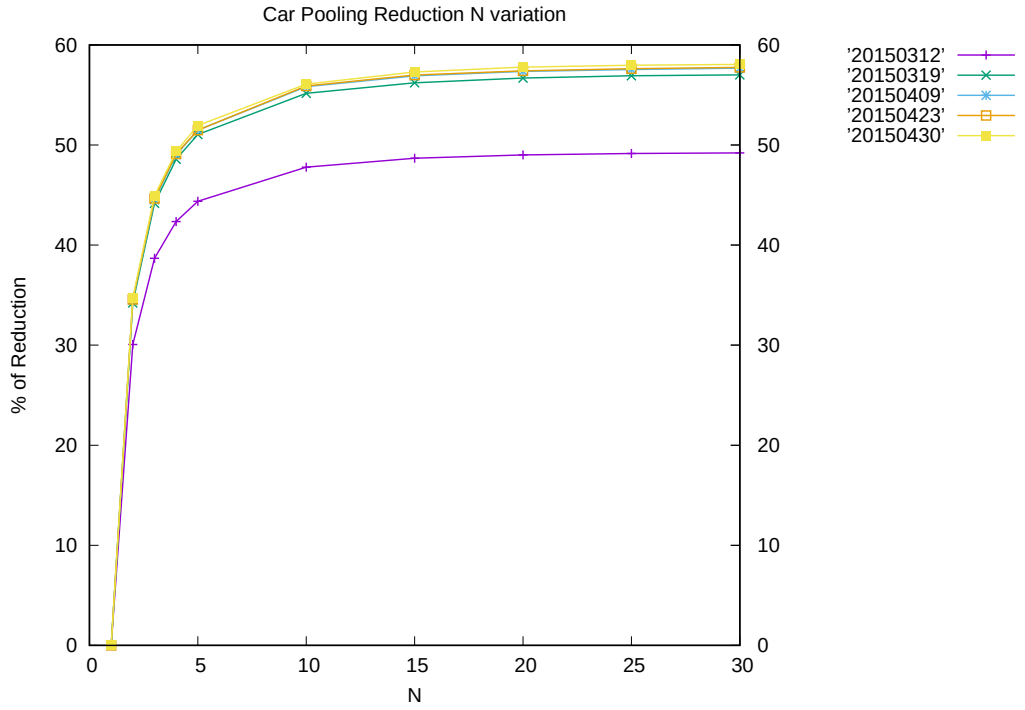

Figura 3.25: Percentuale di riduzione la variare del numero di passaggeri

Conclusioni e Sviluppi Futuri

Concludendo questo lavoro di tesi, si ritiene che il *tool chain* rappresenta uno strumento abbastanza potente per effettuare studi di vario genere sulla mobilità veicolare in quanto il suo punto di forza è la completa genericità data ai moduli realizzati ed infatti abbiamo visto come cambiando un semplice parametro, si possa muoversi da un contesto all'altro. Ad esempio noi ci siamo soffertmati sul caso del *car pooling* prettamente urbano, ma variando tempo e distanza ed ammettendo valori molto più alti da quelli da noi utilizzati, si potrebbe estendere il caso al *car pooling* nazionale effettuando *matching* di veicoli anche tra città diverse.
Allo stesso tempo,

- supponendo di avere i dataset ricalibrati delle tracce di mobilità delle maggiori città italiane o del mondo, si potrebbero estendere i nostri stessi studi realizzati a queste città e se si disponesse delle tracce della mobilità di tutti i veicoli delle città, i risultati sarebbero sempre più precisi arrivando quindi ad analizzare un caso reale.

- si potrebbero studiare casi di *taxi sharing* o si potrebbero fornire nuove analisi in grado di migliorare i servizi offerti da UberPool ([15]) o BlaBlaCar ([12])

- si potrebbe pensare di effettuare degli studi per migliorare i servizi offerti dal *CarSharing* sfruttando, ad esempio, il fatto di poter *clusterizzare* più veicoli appartenenti ad una stessa zona (grazie al variare del parametro `D`), al fine di poter andare a piazzare delle auto che potrebbero essere utilizzate da chi ne ha bisogno, nel punto medio delle coordinate delle città nelle quali sono stati *matchati* i veicoli

- Ecc., ecc.

Quindi, in definitiva, abbiamo realizzato uno strumento molto utile il quale, partendo da dei semplici punti GPS è stato in grado di rielaborare tali coordinate appartenenti ai percorsi di alcuni veicoli della città di Bari e di verificare se è possibile diminuire il numero delle auto circolanti per l'area urbana, sotto determinate condizioni, avendo coì un impatto fondamentale sulla qualità di vita della città, ma anche dei vantaggi a livello ambientale con conseguente diminuizione dell'inquinamento.

Riferimenti Bibliografici

[1] L. Bedogni, M. Gramaglia, A. Vesco, M. Fiore, J. Harri, F. Ferrero, *The Bologna Ringway Dataset: Improving Road Network Conversion in SUMO and Validating Urban Mobility via Navigation Services*, on IEEE Transactions on Vehicular Technology (IEEE TVT), 2015, pp. 1-13

[2] M. Bruglieri, D. Ciccarelli, A. Colornia, A. Luè, *PoliUniPool: a carpooling system for Universities*, Science Direct, Procedia Social and Behavioral Sciences 20 (2011), pp. 558-567

[3] V. Caiati, L. Bedogni, L. Bononi, F. Ferrero, M. Fiore, A. Vesco, *Estimating Urban Mobility with Open Data: A Case Study in Bologna*, IEEE Second International Smart Cities Conference, Trento, Italy, 09/2016.

[4] C. Celes, F.A. Silva, A. Boukerche, R.M.C. Andrade A.A.F. Loureiro, *Improving VANET Simulation with Calibrated Vehicular Mobility Traces*, IEEE Transactions On Mobile Computing, pp. 1-15

[5] R. Meijkamp, R. Theunissen *Car Sharing Consumer Acceptance and Changes on Mobility Behaviour*

[6] S. Fleury, A. Tom, E. Jamet, E. Colas-Maheux *What drives corporate carsharing acceptance? A French study*, Transportation Research Part F, pp. 218-227

[7] A. Furno, M. Fiore, R. Stanica, C. Ziemlicki, Z. Smoreda, *A Tale of Ten Cities: Characterizing Signatures of Mobile Traffic in Urban Areas*, IEEE Transaction on Mobile Computing, pp. 1-14

[8] A. Furno, M. Fiore, R. Stanica, *Joint Spatial and Temporal Classification*

[9] D. Naboulsi, M. Fiore, *Characterizing the Instataneous Connectivity of Large-scale Urban Vehicular Networks*,

[10] D. Naboulsi, M. Fiore, S. Ribot, R. Stanica, *Large-scale Mobile Traffic Analysis: a Survey*

[11] J. Swan, J. Drake, E. Ozcan, J. Goulding, J. Woodward, *A Comparison of Acceptance Criteria for the Daily Car-Pooling Problem*

[12] BlaBlaCar: https://www.blablacar.it/

[13] Car Pooling: https://en.wikipedia.org/wiki/Carpool

[14] Car Sharing: https://en.wikipedia.org/wiki/Carsharing

[15] UberPool: https://www.uber.com/en-IT/ride/uberpool/

Ringraziamenti

Nonostante possa sembrare qualcosa di scontanto, desidero ringraziare le prime persone che mi hanno sostenuto sempre in qualsiasi momento e che mi hanno permesso di raggiungere il mio traguardo seppur composto da difficoltà: i miei genitori.
Ringrazio anche il professor Luciano Bononi per avermi concesso la possibilità di poter lavorare ad un progetto interessante come quello qui realizzato ed il dottor Luca Bedogni per la disponibilità mostrata ogni volta che gli ho chiesto un consiglio.
Dico grazie a mio fratello Carlo per l'esempio che, anche inconsapevolmente, mi fornisce ogni giorno e che oggi avrei voluto avere qui con me.
Dico grazie ai miei "compagni di viaggio" Gianluca e Antonella, a Lara che mi è stata vicina nei miei momenti no, a Emanuele che quando c'è da divertirsi lui c'è spesso.
Dico grazie a quella che negli ultimi tre anni è probabilmente divenuta la mia seconda famiglia, ovvero i miei coinquilini: Ermal, Maicol, Idriss e Jean Jacques con i quali ho condiviso momenti indimeticabili in casa.
Dico grazie ai miei amici Gianfilippo, Marika, Luca, Donato e Claudio i quali, nonostante la distanza che condiziona il fatto di poter condividere alcuni giorni insieme, hanno permesso di sentirmi vicino a loro anche se a volte posso essere sembrato assente.
Ringrazio mio cugino Giuseppe ed il mio amico Riccardo per essersi resi disponibili nel momento del bisogno e che hanno deciso di condividere questo momento con me. Desidero ringraziare mio zio Vincenzo per essere qui con me oggi ed i miei nonni, tutti per essersi interessati a me ed aver gioito dei miei successi.