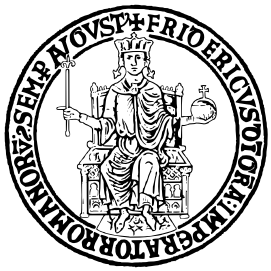
Università degli Studi di Napoli Federico II



Scuola Politecnica e delle Scienze di Base

Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione

Corso di Laurea Triennale in Informatica

Valutazione dell'Efficacia di Strategie di Parking Guidance Attraverso Simulazioni Microscopiche

Relatori
Professor Sergio Di Martino Di Martino
Dott. Luigi Libero Starace Starace

Candidato
Riccardo Converso
N8600/2934

Anno Accademico 2021–2022

Università degli Studi di Napoli Federico II
Scuola Politecnica e delle Scienze di Base
Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione
Corso di Laurea Triennale in Informatica

Valutazione dell'Efficacia di Strategie di Parking Guidance Attraverso Simulazioni Microscopiche

Relatori
Professor Sergio Di Martino Di Martino
Dott. Luigi Libero Starace Starace

Candidato
Riccardo Converso
N8600/2934

Anno Accademico 2021–2022

Indice

Introduzione

Con il rapido sviluppo dell'economia, il tenore di vita delle persone continua a migliorare, il che ha esacerbato l'aumento dei veicoli nei centri urbani. L'aumento del numero delle auto ha facilitato gli spostamenti delle persone e promosso la crescita economica, tuttavia, con il continuo aumento del numero di autoveicoli nelle città, il problema dei parcheggi difficoltosi sta diventando sempre più importante[1].

La ricerca di un posto auto in cui parcheggiare è un problema molto delicato in un gran numero di città, le cause possono essere molteplici, come la scarsa presenza di aree di parcheggio o il continuo aumento di veicoli nelle strade. Cercare parcheggio, oltre ad essere fonte di stress per i guidatori, può portare anche all'aumento del traffico poichè i veicoli essendo in continua ricerca di un' area in cui sostare tenderanno a riempire le strade e di conseguenza costringere all'attesa anche i guidatori che in quel momento non hanno intenzione di parcheggiare. Oltre ai problemi elencati precedentemente si può prendere in considerazione anche la componente ambientale, poichè l'aumento del traffico comporta inevitabilmente un aumento degli indici di inquinamento e un notevole consumo di carburante per le singole vetture.

La rapida digitalizzazione della vita quotidiana di ogni individuo ha portato, nel corso degli anni, a risolvere gran parte delle problematiche che giornalmente una persona deve affrontare. Il 67.1% della popolazione mondiale disponde di uno smartphone e il 62.5% è connesso ad internet [2], questi dati ci forniscono un'istantanea del mondo che ci circonda e di come le persone sfruttano i propri dispositivi tecnologici come supporto essenziale per migliorare la propria vita.

Nell'ambito del problema del parcheggio, sfruttare tecnologie in grado di poter ridurre in qualche modo il traffico presente nelle città più popolose , potrebbe interessare non solo il guidatore stesso, ma anche i singoli comuni delle varie città. Una delle tante applicazioni tecnologiche che si possono trovare in questo ambito può essere quella di informare i conducenti su dove trovare un parcheggio corrispondante alle loro esigenze, evitando quindi il vagare per minuti alla ricerca di un posto.

Tali tecnologie prendono il nome di sistemi di Parking Guidance, essi possono essere di vario tipo a seconda dell'informazione che essi forniscono. Essi possono dare informazioni riguardo, ubicazione, capacità o dimensione dei posti auto.
L'applicazione nel mondo reale è possibile tramite telecamere postate accanto ai parcheggi su strada o con l'utilizzo di appositi sensori. Utilizzare queste tecnologie può risultare molto dispendioso per il comune di una cittàe inoltre hanno bisogno di una continua manutanzione. Di conseguenza, al fine di poter giustificare tale investimento è possibile effettuare, come in questo studio, diverse simulazioni per poter calcolare i risultati ottenuti dall'utilizzo dei PGS (Parking Guidance Systems). Lo studio presentato in questo elaborato andrà a quantificare i benefici escaturiti dall'utilizzo di sistemi di Parking Guidance, sfruttando il simulatore SUMO. Sulla mappa della Business-Area di San Francisco verranno presentate diverse strategie che simuleranno il comportamento dei conducenti in diversi scenari di traffico. Il risultato finale sarà quello di quantificare gli eventuali benefici che comporterà la scelta di applicare sistemi altamente tecnologici per la guida ai parcheggi.

Capitolo 1

Smart Mobility e Parking Guidance

1.1 Sistemi di Parking Guidance PGS

Trovare un posto libero per parcheggiare nei centri urbani è un notevole problema di mobilità sopratutto se parliamo di città con una densità di popolazione molto elevata e la mancanza di informazione da parte dei conducenti su dove trovare un parcheggio libero implica inevitabilmente un aumento del traffico. Per risolvere questo problema entrano in gioco sistemi di Parking Guidance che hanno il compito di informare il guidatore riguardo le aree di sosta o parcheggi dove hanno maggiori probabilità di trovare posti liberi.
"I Parking Guidance Sistems combinano le tecnologie di monitoraggio del traffico, comunicazione, elaborazione e segnaletica per fornire ai conducenti informazioni sul parcheggio all'interno di aree controllate" [3].

Numerosi studi affermano che, grazie ai PGI (Parking Guidance Information), la probabilità di trovare parcheggi disponibili è significativamente aumentata, e anche il tempo di ricerca di ogni guidatore si è ridotto notevolmente [4]. I PGI forniscono un gran numero di informazioni riguardo la quantità di parcheggi presenti nelle vicinanze e la loro ubicazione, inoltre possono fornire anche informazioni riguardo la tariffa delle aree di sosta a pagamento così da poter dare la possibilità all'utente di poter scegliere, in base a vari parametri, il parcheggio in cui sostare.
In questo modo si possono evitare situazioni in cui i veicoli circolano a vuoto in cerca di un parcheggio, trovando rapidamente un posto libero, l'obiettivo è quello di ridurre i tempi di ricerca, che a loro volta alleviano la congestione del traffico, con la relativa riduzione dell'inquinamento.

Tuttavia l'utilizzo di tali sistemi di Parking Guidance sono ancora molto limitati per

quanto riguarda il raccogliere informazioni dettagliate sulla disponibilità dei parcheggi in tempo reale, in particolare sopratutto riguardo i parcheggi su strada [4].
Per questo motivo possiamo individuare tre tipologie principali di informazioni riguardo i parcheggi su strada: Informazione Statica, Informazione Dinamica e Previsione della disponibilità dei posti auto.

1.1.1 Informazione Statica

L'informazione di tipo statico si basa su dati costanti che non subiscono variazioni in tempo reale e che possono essere aggiornati a periodi regolari in linea con i cambiamenti che può subire il disegno dell'area urbana o l'aggiunta di nuovi posti auto.
I dati che i sistemi di Parking Guidance offrono riguardano l'ubicazione esatta dei posti auto su strada, compresa di capacità o tariffa oraria da pagare per poter sostare in quella determinata zona della città. Il fine è offrire al guidatore l'intero elenco di parcheggi vicini al suo punto di destinazione così che non dovrà circolare a vuoto senza conoscere la reale posizione e quantità di parcheggi presenti.

Tali informazioni sono generalmente fornite dal comune che può riportare anche dati relativi alla capacità di un parcheggio su strada, tale informazione si rierisce solo al numero di veicoli che può contenere quel parcheggio, non tenendo conto che la capacità può cambiare nel tempo.
I motivi per cui la capacità di un parcheggio può subire variazioni nell'arco di pochi minuti sono la guida nelle ore di punta, lavori in corso lungo la carreggiata, la dimensione dei veicoli parcheggiati e la distanza tra essi [5].

1.1.2 Informazione Dinamica

Questo tipo di informazione consiste nel fornire dati in tempo reale della disponibilità di parcheggio in strada. Sfruttando tali informazioni i conducenti possono sapere in tempo reale se nella propria zona di destinazione troveranno posti liberi in cui parcheggiare e tutto ciò è possibile grazie all'installazione di sensori lungo le aree di parcheggio. Con l'ausilio di sensori, in questo modo, nell'istante in cui un veicolo abbandonerà il parcheggio, gli altri conducenti saranno notificati del posto che si è appena liberato.
Gli utenti, sapendo che in una determinata zona non sono presenti parcheggi liberi in quel momento, andranno diretti nelle aree in cui ci sono posti disponibili. Tra i sensori che possono essere utilizzati in questo ambito ci sono i magnetometri, strumenti di misura del campo magnetico che possono essere installati nell'asfalto e che possono rilevare la presenza di un veicolo in prossimità [6].

Questo tipo di tecnologia è altamente affidabile ed efficiente, con la possibilità di offrire aggiornamenti continui in tempo reale, ma di contro l'installazione e la manutenzione di sensori di questo tipo possono generare un costo eccessivamente elevato per i singoli comuni.
Una soluzione a questo contro appena descritto è rappresentato dall'utilizzo di app per il monitoraggio e il tracciamento dei veicoli. Questa soluzione consiste nel fornire un'app che può essere scaricata dagli utenti dalla quale viene stimata l'ubicazione di tutti i veicoli in quella zona tramite il GPS. In questo modo è possibile capire quanti veicoli sono presenti in quella zona in quell'istante e quanti sono parcheggiati nelle aree di sosta.
Di contro però tale tecnologia non è ancora realizzabile poichè è necessario che tali app siano installate da un elevato numero di utenti affinchè possano essere al cento per cento efficienti e applicazioni in questione non sono ancora ampiamente diffuse [7].

1.1.3 Previsione Disponibilità Posti Auto

Questo tipo di informazione si basa sull'effettuare un pronostico riguardo la disponibilità dei parcheggi. Quando un veicolo lascia un parcheggio verrà registrato al fine di poter realizzare uno storico del numero di veicoli che transitano in quella strada e di quanti saranno parcheggiati in quella zona in una determinata fascia oraria.
Questo tipo di informazione è altamente dinamica poichè consiste nello sfruttare lo storico dello stato di tutti i veicoli che in quella fascia oraria si trovano in quella data area, ed effettuare una predizione o un calcolo di probabilità di trovare un parcheggio libero [8].

1.2 I PGS oggi

I PGS e la loro applicazione si è evoluta molto negli ultimi anni, sfruttando tecnologie a ultrasuoni o di sensoristica forniscono informazioni sulla disponibilità dei posti auto e allo stesso tempo, le nuove tecnologie basate su telecamere, consentono di leggere la targa dei veicoli in ogni parcheggio. Si tratta di un valore aggiunto in quanto consente di identificare un determinato veicolo in una specifica area di sosta e, inoltre, registra eventuali incidenti occorsi in tale spazio. Questi nuovi metodi, oltre a rendere l'esperienza di guida migliore, aumentano la sicurezza durante la fase di uscita o entrata in un parchegio. Un utilizzo standard di sensori di parcheggio è presentata in [9], dove dopo l'analisi e l'elaborazione dei dati ottenuti sfruttando una rete di sensori wireless

con topologia ad albero, il centro informazioni e gestione distribuisce le informazioni sul parcheggio tramite schermi a LED e display per i conducenti in tempo reale.
In Italia, l'utilizzo di tecnologie avanzate è dedicato principalmente ai parcheggi custoditi a pagamento o a garage. L'utilizzo di app per la prenotazione dei parcheggi è molto diffuso grazie al facile controllo e gestione dei posti auto nelle aree private e, essendo facilmente reperibili, sono altamente utilizzate sopratutto da turisti o da automobilisti non pratici della città. Queste applicazioni indicano, su richiesta dell'utente, quali parcheggi privati si trovano nelle proprie vicinanze, indicando informazioni relative al costo, alla presenza di posti liberi o altro.
L'utilizzo di sensori presenti nell'asfalto o di telecamere installate in prossimità dei parcheggi possono comportare un costo molto elevato e una manutenzione frequente, una soluzione più economica che in detereminate situazioni può risultare efficace è l'utilizzo di app apposite.
Un esempio può essere quello di Spotter [10], un'app in cui gli utenti segnalano tramite il loro cellulare di aver liberato il proprio posto auto, così da segnalarlo all'intera community interessata. Il limite di quest'applicazione sta nella sua scarsa popolarità che non garantisce un tracciamento completo di tutta l'area urbana e sopratutto nella mancata attenzione degli utenti nel segnalare di aver lasciato il parcheggio in quel momento.
Per ovviare a quest'ultimo problema, Spotter sta studiando un sistema basato su ricompense o punti da conferire agli utenti che tenderanno a segnalare un gran numero di parcheggi liberi.
L'utilizzo di PGS basati su sensori sono utilizzati in diverse città nel mondo come New York, San Francisco o Londra. In questo ambito i ricercatori della Rutgers University del New Jersey hanno studiato un sistema che si basa sull'inserire dei sensori a basso costo su ogni taxi della città, in questo modo girando per l'intero centro urbano rilevano tutte le aeree libere. Le informazioni ottenute dai sensori vengono combinate con le coordinate ottenute dai GPS per poi essere ditribuite sugli smartphone degli utenti [11].

1.3 Quantificare i vantaggi dei PGS

Nel paragrafo precedente sono state elencati i diversi tipi di informazione che i Parking Guidance System possono offrire, specificando sopratutto i grandi vantaggi che possono portare al fine di poter migliorare la fase di ricerca del parcheggio dei singoli conducenti. Nonostante i fattori positivi che possono portare tali tecniche, ci sono aspetti che

possono rendere di difficile realizzazione l'applicazione di tali sistemi.
Gli alti costi e la complessità dell'installazione di tali tecnologie rendono difficile il quantificare i reali benefici o gli eventuali svantaggi che può comportare la scelta di adottare o non adottare sistemi di Parking Guidance.

I vantaggi effettivi per il conducente di poter aver accesso a informazioni sulla disponibilità dei parcheggi in tempo reale è stato studiato a malapena, e ad oggi, esistono solo indagini basate su dati simulati [12]. Per quanto riguarda lo studio del comportamento dei conducenti che non sono a conoscenza della reale ubicazione e disponibilità dei parcheggi sono stati effettuati degli studi su simulatori [13].
Dallo studio effettuato in [13] è stato sfruttato un modello che simula il comportamento di un conducente in un ambiente urbano, tale simulazione è in grado di catturare le complesse dinamiche di auto-organizzazione delle grandi città all'interno di uno spazio (stradale) realistico. Il modello considera diversi parametri come tempo di ricerca, distanza percorsa e costi di parcheggio su diversi gruppi di conducenti.

Per stimare la quantità e i benefici relativi all'adozione di sensori in prossimità dei parcheggi sulla base dei dati nel mondo reale in [14] è stato simulato uno scenario casuale ed è stata misurata la durata della fase di ricerca di un parcheggio fino a quando non è stato trovato il primo posto libero.
La città di San Francisco ha sperimentato e documentato con cura un ampio e innovativo programma di gestione dei parcheggi chiamato *SFpark*, lo studio in [14] utilizza i dati ottenuti per poter effettuare le proprie analisi. In primo luogo tenteranno di stabilire i criteri qualitativi che caratterizzano un parcheggio, come capacità ubicazione o dimensione. In secondo luogo, forniscono una valutazione dell'efficacia di SFpark, esaminando gli impatti sul comportamento dei conducenti del programma nel suo insieme.

Precedentemente è stato stabilito che applicare sistemi di Parking Guidance può essere molto dispendioso in termini economici quindi la domanda a cui bisogna rispondere è quanto effettivamente può convenire investire su determinate tecnologie e in quali tipi di scenari urbani è preferibile adottare una rete basata su sistemi statici o su sistemi dinamici.

1.3.1 Classificazione dei centri urbani

I centri urbani che possono essere presi in analisi non possono essere trattati tutti allo stesso modo poichè la richiesta di parcheggio aumenterà in base a determinati fattori che, adottando sistemi di Parking Guidance, possono far evincere risultati differenti. Una classificazione in base al tempo impiegato in media dai conducenti per trovare

un posto auto nelle principali città europee e americane si può trovare in [15], dove a seconda di diversi parametri come la densità di popolazione, la dimensione del centro urbano e il tenore di vita delle persone si possono evincere grandi differenze.

Si può dedurre come nelle città in cui il centro urbano è molto distante dalle zone residenziali il tempo speso per cercare parcheggio dei singoli conducenti sia più elevato. Per quantificare in termini numerici basta osservare che negli USA un guidatore spende in media circa 17 ore all'anno per cercare parcheggio e in termini economici costa 345$ per guidatore solo per carburante ed emissioni [16]. Lo scenario italiano è anch'esso preoccupante, dove gli utenti impiegano in media 30 minuti al giorno in cerca di un posto auto [17].
La soluzione rappresentata dall'utilizzo dei PGS è in linea teorica molto valida, il lato economico che scaturisce dall'utilizzo di nuove tecnologie è un ostacolo concreto per molti e i benefici non sono facilmente quantificabili.

Capitolo 2

Tecnologie Utilizzate

Per rispondere al problema presentato nel Capitolo 1, in questo studio sono state utilizzate diverse tecnologie , il simulatore SUMO [18], l'interfaccia TraCI [19], script in Python [20] e le mappe ottenute da OpenStreetMap [21].

2.1 SUMO - Simulation of Urban MObility

Simulation of Urban MObility (Eclipse SUMO) è un software di simulazione del traffico multimodale open source, portatile e microscopico progettato per gestire reti di grandi e piccole dimensioni. Consente la simulazione intermodale, inclusi i pedoni, e viene fornito con un ampio set di strumenti per la creazione di scenari [18]. SUMO offre la possibilità di effettuare simulazioni di scenari microscopici in cui è possibile gestire il comportamento di tutti gli agenti che compongono uno scenario stradale, quali pedoni, veicoli o trasporto pubblico.
Dopo aver simulato uno scenario è possibile recuperare l'intero dataset generato dalla simulazione, che può comprendere, distanza percorsa dai veicoli, collisioni verificate durante tutta la simulazione, indici di traffico, emissioni e tanto altro. Per svolgere lo studio presentato in questo elaborato è stato preferito SUMO rispetto ad altri simulatori poichè , oltre ad essere opensource, offre una ricca interfaccia grafica eseguibile anche su macchine non altamente prestanti e, utilizzando *NetEdit* rende la preparazione degli scenari estremamente semplici e intuitivi da realizzare.
Facendo una breve panoramica dei servizi offerti da SUMO si può partire dall'editor *NetEdit* con una breve descrizione degli agenti del traffico che possono essere creati e gestiti con il simulatore.

2.1.1 NetEdit

NetEdit è un editor di reti stradali offerto da SUMO, con tale strumento è possibile costruire scenari da zero, con la possibilità di inserire strade, corsie, parcheggi, veicoli o pedoni. Tramite NetEdit, una volta costruita una rete stradale, è possibile evidenziare i percorsi che i singoli veicoli possono intraprendere per poi verificare il risultato finale importando la rete appena creata in SUMO.
Entrando nello specifico, SUMO accetta dei files *.xml* nei quali è inserito l'intero scenario da simulare, per esempio il numero di veicoli da generare con le loro caratteristiche. Utilizzando NetEdit è possibile creare i files *.xml* necessari per la simulazione, senza dover scrivere codice ma sfruttando la sola interfaccia grafica offerta.

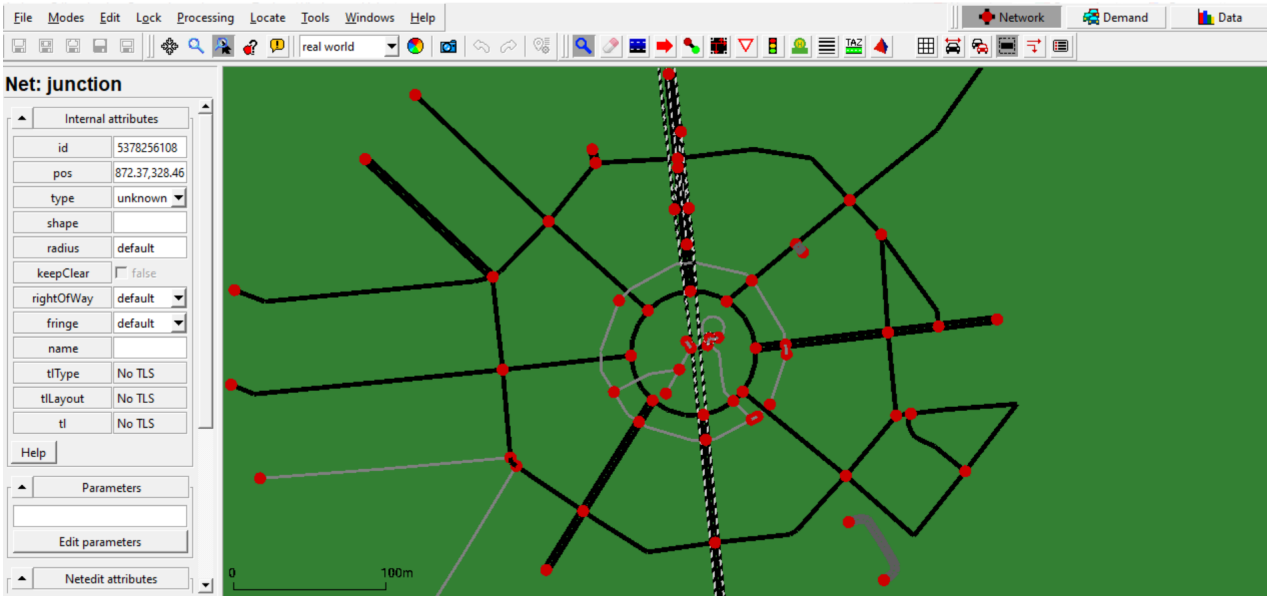


Figura 2.1: Esempio di rete stradale costruita in NetEdit

La Figura 2.1 mostra l'interfaccia presentata da NetEdit, essa rappresenta una rete stradale composta da strade *(edges)* che uniscono i vari punti *(junction)* in cui è possibile generare la domanda di traffico nell'intera mappa.
Due punti in SUMO possono essere collegati da una strada, che a sua volta può essere suddivisa in corsie *(lane)*, le quali possono essere orientate come preferisce l'utente. Una volta costruita una rete stradale bisogna comprendere che qualsiasi elemento presente nella mappa ha bisogno di un identificativo per poter essere riconoscibile, qundi il passaggio successivo consiste nel passare alla *modalità ispezione* e modificare gli id degli oggetti.

Una strada può avere diversi attributi che possono essere modificati dall'utente, in Figura 2.2 si notano i campi *id* che rappresenta l'identificativo della strada, *from* che rappresenta la posizione di partenza della strada, *to* la posizione di fine della strada, *speed* il limite di velocità in m/s, *priority* un numero che determina la priorità tra i diversi tipi di strada poichè SUMO deriva le regole del diritto di precedenza agli incroci dalla priorità data. Tra questi parametri si può trovare anche *allow* e *disallow*, questi due campi rappresentano i tipi di veicoli che possono circolare su quella specifica strada. In questo modo si può distinguere una corsia pedonale da una ciclabile o per esempio, una corsia per gli autobus dalle rotaie di un tram. SUMO offre numerose classi di veicoli che possono essere utilizzati, e permette inoltre di poterne modificare o crearne di nuovi [Figura 2.3].

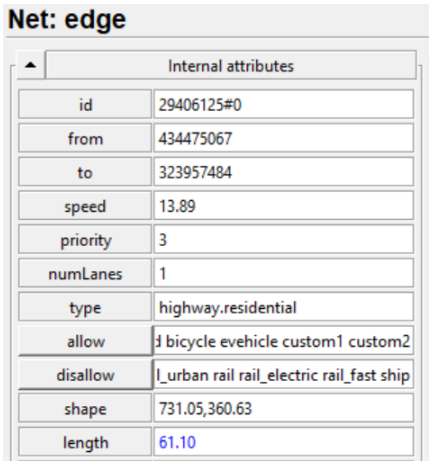


Figura 2.2: Attributi di una strada

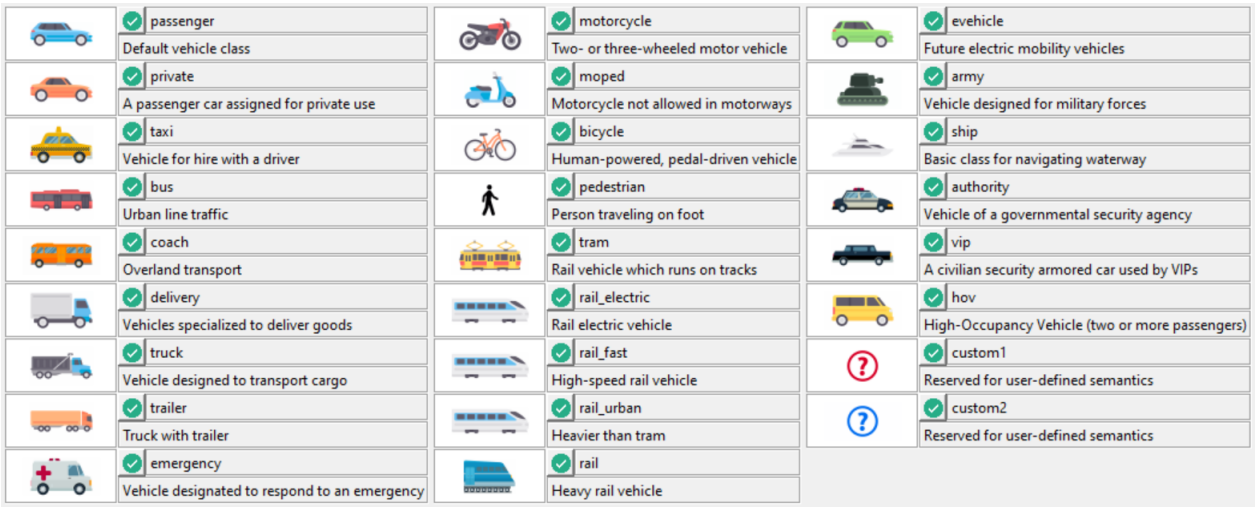


Figura 2.3: Classi di veicoli in SUMO

Per quanto riguarda i veicoli creati dagli utenti, presentano un parametro *vClass* che identifica il tipo di veicolo da creare, mentre gli altri parametri riguardano la velocità che può raggiungere, la dimensione del veicolo o consumo di carburante.

2.1.2 Gli "Additionals"

Nel paragrafo precedente sono stati introdotti gli elementi fondamentali che compongono la rete stradale quali strade, corsie e veicoli. In SUMO è possibile anche inserire oggetti chiamati *additionals*, che incidono sul comportamento dei veicoli. Tra gli additionals si trovano semafori, fermate degli autobus, fermate dei taxi, parcheggi e tanto altro.
Il comportamento dei veicoli sarà influenzato dalla presenza degli additionals, questi oggetti presenteranno anch'essi dei parametri modificabili. Un semaforo potrà essere programmato in base al tipo di comportamento che si vuole far gestire, può essere impostata la durata dei vari colori che può assumere e quali tipologie di veicoli dovranno obbligatoriamente fermarsi, per esempio si potrebbe impostare che le ambulanze non saranno costrette a fermarsi quando il semaforo sarà rosso. Per quanto riguarda le fermate dei pullman, SUMO permette di poter creare non solo singoli pullman che circolano per l'area urbana, ma anche vere e proprie linee degli autobus con tanto di percorsi pre-stabiliti che dovranno seguire.
Le fermate degli autobus avranno quindi un ruolo centrale in cui, se durante il percorso il pullman incontrerà una fermata dovrà fermarsi per permettere ai pedoni di scendere o salire dal bus.

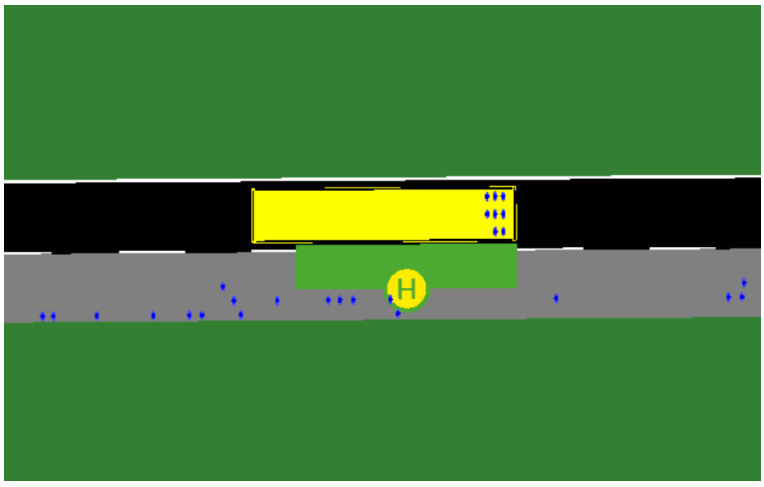


Figura 2.4: Esempio di bus in SUMO

2.1.3 I Parcheggi

I parcheggi sono degli addictionals, e sono rappresentabili sia come parcheggi su strada sia come vere e proprie aree di sosta. Un parcheggio si presenta al lato di una strada [Figura 2.5] e, i veicoli che nel loro percorso prevedono di dover parcheggiare, si fermeranno in prossimità dell'area di sosta iniziando la manovra di entrata nel posto auto.

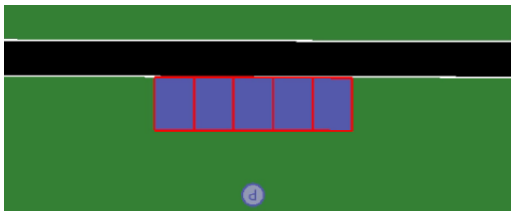


Figura 2.5: Parcheggio in SUMO

I parcheggi in SUMO, come qualsiasi elemento che compone una rete stradale, presentano diversi parametri che permettono di modificare il parcheggio seguendo le esigenze dell'utente.

In Figura 2.6 sono rappresentati i parametri che possono essere modificati tra cui in *lane* verrà inserito l'id della corsia in cui si trova il parcheggio, *startPos* e *endPos* sono rispettivamente posizione in cui inizia e termina il parcheggio, *roadsideCapacity* è la capacità del parcheggio infine *width*, *length* e *angle* rappresentano la dimensione e l'angolo del parcheggio. Modificando questi parametri è possibile costruire parcheggi di ogni tipo, di grandi dimensioni, parcheggi singoli, parcheggi paralleli alla strada o a "spina di pesce".
Un parametro fondamentale per l'ambito in cui è stato utilizzato il simulatore SUMO è *roadsideCapacity*, poichè tale campo è determinante per distinguere le diverse strategie che sono state studiate, esso deve essere visto come capacità del parcheggio dal punto di vista statico, ovvero la capienza massima di veicoli che può ospitare.

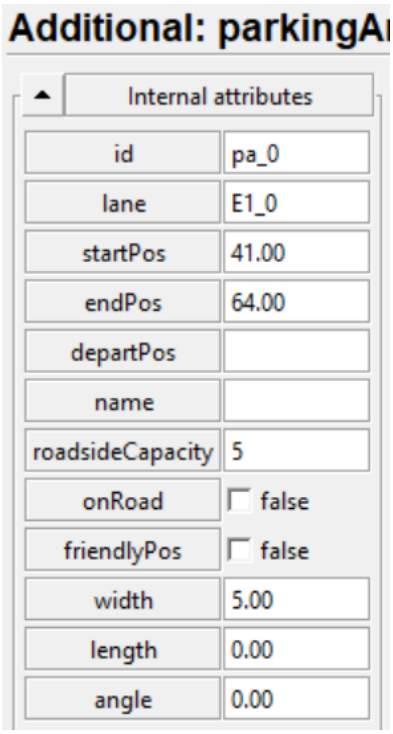


Figura 2.6: Attributi di un Parcheggio

2.1.4 Rerouting e ParkingRerouting

In questo paragrafo verranno trattati dei dispositivi fondamentali per la realizzazione dei differenti scenari in SUMO. Un rerouter è un dispositivo che si colloca nelle corsie (scelte dall'utente) in cui i veicoli, nell'istante in cui attraverseranno tali sistemi, modificheranno il proprio comportamento a seconda di ciò che è stato impostato dall'utente. Per definizione, la sua utilità è cambiare il percorso di un veicolo non appena esso si sposta su una delle aree contrassegnate da un Rerouter. Per esempio, considerando una rete stradale che contiene due *lane* chiamate rispettivamente A e B, un utente può decidere di aggiungere un Rerouter in prossimità della lane A e impostarlo in modo che tutti i veicoli che lo attraverseranno dovranno modificare la propria destinazione in favore della lane B.

Anche un Rerouter presenta diversi parametri come in Figura 2.7, escludendo i campi già descritti nei paragrafi precedenti possiamo notare *edges* che rappresenta gli id delle strade in cui è presente tale Rerouter, *probability* è un numero compreso tra 0 a 1 che identifica la probabilità che un veicolo possa essere reindirizzato in una nuova destinazione, infine *timeThreshold* è il tempo di attesa minimo prima che il reindirizzamento abbia effetto. Un reindirizzamento può funzionare in diversi modi. Entro un periodo di tempo è possibile chiudere una strada, assegnare nuove destinazioni o percorsi predefiniti ai veicoli.

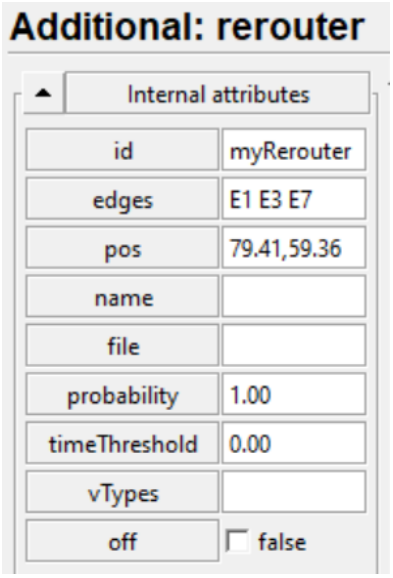


Figura 2.7: Attributi di un Rerouter

I Rerouter sono uno strumento molto importante e la loro utilità non si restringe solo al campo del reindirizzamento dei veicoli verso nuove destinazioni, bensì è possibile sfruttare tali dispositivi anche per stabilire una politica di ricerca di parcheggi da parte dei veicoli. In questo caso parleremo di *"ParkingRerouting"*, il modo in cui funzionano è simile a quello visto con i Rerouting ma con la differenza che i veicoli non verranno reindirizzati verso nuove strade, bensì verso nuovi parcheggi da visitare.

Lo scenario è quello in cui un veicolo, intento a parcheggiare, nel caso in cui dovesse trovare un parcheggio con nessun posto libero, in mancanza di ParkingRerouting attenderà un tempo indefinito accanto ad esso finchè uno dei veicoli parcheggiati non avrà terminato la propria sosta [Figura 2.8].

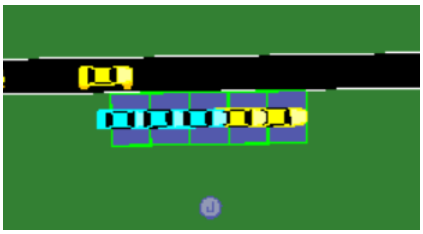


Figura 2.8: Comportamento veicolo in mancanza di ParkingRerouting

In presenza invece di un dispositivo di ParkingRerouter i veicoli, se dovessero incontrare un parcheggio completamente occupato, verranno reindirizzati verso un nuovo parcheggio. Il reindirizzamento verso aree di parcheggio alternative si occupa di definire un insieme di aree di parcheggio che possono essere utilizzate reciprocamente come alternative. Esso viene attivato in due casi in particolare, quando un veicolo raggiunge un'area di parcheggio e non è in grado di parcheggiare poichè il parcheggio è pieno, quando la destinazione di un veicolo è verso un parcheggio con l'attributo *visible="true"* e il parcheggio in quel momento risulta pieno.
L'attributo *visible* è fondamentale per il fine dello studio effettuato, poichè rende visibile a tutti i veicoli presenti nella mappa tutti gli attributi di quel parcheggio, compresa la propria disponibilità. Quindi si può immaginare come se tutti i parcheggi con visibilità impostata su *"true"*, che siano stati integrati con una tecnologia di ParkingGuidance, permettono ai conducenti di conoscere in tempo reale la disponibilità di quei posti auto. Con SUMO è possiblie ampliare il concetto di reindirizzamento verso un'area di parcheggio alternativa grazie all'ausilio di algoritmi già presenti nel simulatore con i quali è possibile costruire scenari differenti [Figura 2.9]. Tra gli algoritmi di reindirizzamento più utilizzati ci sono, *parking.probability.weight* in cui la probabilità di scegliere un parcheggio piuttosto che un altro è condizionata dal parametro *probability* presente tra gli attributi di un ParkingRerouting, *parking.capacity.weight* dove i veicoli sceglieranno il prossimo parcheggio da visitare scegliendo quello con la capacità più alta, oppure *parking.absfreespace.weight* in cui i veicoli saranno reindirizzati verso il parcheggio con più posti liberi.

Parameter Name	Description
parking.probability.weight	the influence of the *probability* attribute of the `parkingAreaReroute`
parking.capacity.weight	The total capacity of the parking area
parking.absfreespace.weight	The absolute number of free spaces
parking.relfreespace.weight	The relative number of free spaces
parking.distanceto.weight	The road distance to the parking area
parking.timeto.weight	The assumed travel time to the parking area
parking.distancefrom.weight	The road distance from the parking area to the vehicles destination
parking.timefrom.weight	The assumed travel time from the parking area to the vehicle destination

Figura 2.9: Da documentazione di SUMO [18] algoritmi di rerouting

2.1.5 I Detcetor

I detector sono dei dispositivi che fanno parte della famiglia degli *Addictionals* e hanno l'utilità di segnalare i veicoli e il loro stato nel momento in cui verranno attraversati.
I dati ottenuti dai dispositivi sono accessibili tramite dei file di output che possono essere generati seguendo le esigenze dell'utente. I dati possono riguardare l'istante in cui è stato attraversato un detector, la velocità in quell'istante, il quantitativo di CO2 emesso e tanti altri valori. Tramite TraCI è possibile accedere ai dati descritti precedentemente per poter ottenere informazioni che possono essere utilizzate al fine di modificare il comportamento di determinati veicoli in un certo stato. Con l'utilizzo dei detectors si può inoltre modificare il comportamento dei veicoli durante la simulazione, TraCI offre la possibilità, utilizzando apposite funzioni, di cambiare destinazione ai veicoli, cambiare corsia, diminuire la velocità, far fermare un veicolo o indirizzarlo verso un nuovo parcheggio.

2.2 TraCI

TraCI è l'abbreviazione di "Traffic Control Interface". Dando accesso ad una simulazione del traffico stradale in corso, permette di recuperare i valori degli oggetti simulati e di manipolarne il comportamento durante la simulazione stessa [Figura 2.10].
TraCI utilizza un'architettura client/server basata su TCP per fornire l'accesso a SUMO [18]. Una simulazione eseguita con TraCI viene lanciata da riga di comando eseguendo un apposito script in Python [20], infatti sfruttando tali script è possibile modificare il

comportamento dei veicoli nel corso della simulazione stessa, al termine di essa, è possibile generare diversi files *.xml* nei quali sono presenti tutti i dati che possono essere utili per fini di studio. I comandi di TraCI si suddividono quindi in due macro-categorie: *Value Retrieval* e *State Changing*.
Le due categorie citate sono degli insiemi di funzioni che durante o al termine della simulazione possono essere lanciate o per ottenere dati utili o per modificare il comportamento dei veicoli. Di seguito verranno elencati gli utilizzi e le funzioni più importanti offerte da entrambe le due categorie.

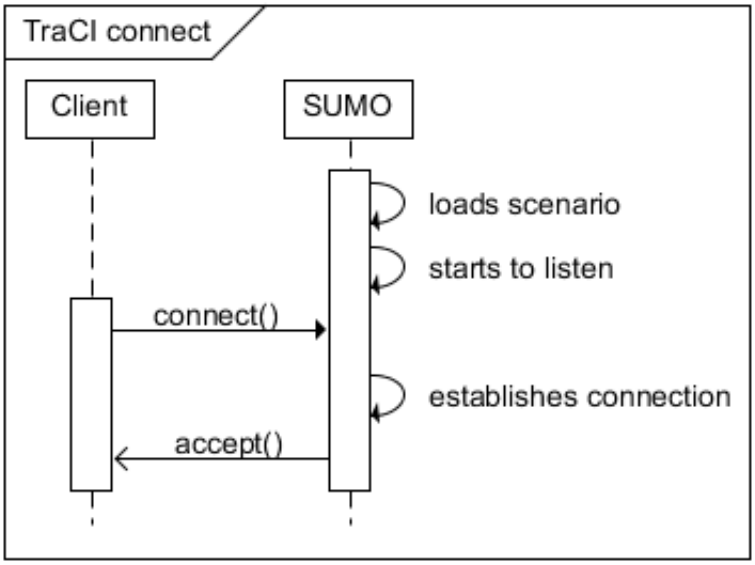


Figura 2.10: Da documentazione di SUMO [18] Protocollo TraCI

2.2.1 Value Retrieval

Questa categoria si riferisce alla possibilità di poter ottenere dati dalla simulazione che possono essere elaborati sia a fine che durante la simulazione stessa, per poter analizzare il comportamento dei singoli veicoli. Le funzioni che compongono tale categoria si possono riferire a diversi elementi della simulazione, per esempio dati riguardo il traffico, riguardo la rete stradale o riguardo i dispositivi su strada (come i Rerouting). È possibile ottenere il numero dei veicoli presenti in quell'istante di tempo nella simulazione, ottenere la velocità di un detereminato veicolo, calcolare le emissioni di CO2, il consumo di carburante e tanto altro.

Tra le funzioni utilizzate nell'ambito dello studio svolto in questo elaborato troviamo, *isStoppedParking* utilizzata per poter determinare se un dato veicolo è fermo in un parcheggio, oppure *getNextStops* che genera l'elenco di tutti i prossimi parcheggi che un veicolo può incontrare lungo il suo tragitto, sempre se non verrà reindirizzato verso un'altra destinazione.
Un altro modo per ottenere dati da una simulazione senza l'utilizzo di TraCI è tramite la generazione di files di output. SUMO, infatti offre la possibilità, tramite dei comandi che si possono inserire nel file di configurazione della simulazione, di generare file di output *.xml* di ogni genere. Si possono ottenere files relativi alle emissioni, al tragitto, ai parcheggi visitati, al carburante consumato di ogni veicolo della simulazione. Per visualizzare la lista completa dei files di output che si possono generare si rimanda alla documentazione ufficiale di SUMO [18].

2.2.2 State Changing

In questa categoria sono presenti tutte le funzioni che permettono di modificare il comportamento dei singoli veicoli durante la simulazione. Con l'ausilio di queste funzioni, per esempio, si può imporre ai veicoli di rallentare o aumentare di velocità in determinate condizioni, cambiare corsia, cambiare destinazione o parcheggio di destinazione e tanto altro.
Al fine di comprendere al meglio l'importanza di questo strumento si può fare un esempio, supponiamo di avere un veicolo che è intento a cercare un parcheggio libero con scarsi risultati, è possibile programmare un algoritmo sfruttando TraCI che impone al veicolo di allargare la propria ricerca sistematicamente nel momento in cui incontrerà n parcheggi occupati. TraCI rappresenta sicuramente una soluzione a molti problemi e permette di rendere le simulazioni ancora più personalizzabili di quanto è possibile già con SUMO. L'unico limite che può rendere la scelta di utilizzare questo tipo di tecnologia più difficile è relativo all'aumentare del peso della simulazione in termini di elaborazione dei dati per ogni step. Aggiungere alla simulazione una o più funzioni, che in tempo reale, cercheranno di ottenere dati dallo scenario e modificare il comportamento dei veicoli può causare un eccessivo rallentamento della simulazione. Prima di utilizzare TraCI bisognerebbe prima comprendere se è effettivamente necessario o se SUMO non offre già il servizio di cui si necessita.

2.3 Python

Python è un linguaggio di programmazione orientato agli oggetti, interpretato e quindi non ha bisogno di compilazione. I vantaggi relativi a questo linguaggio di programmazione sono la sua semplicità, la sua leggibilità e la sua portabilità.
Essendo un linguaggio open-source è possibile utilizzarlo in qualsiasi scenario e presenta una curva di apprendimento decisamente moderata. Per quanto riguarda l'utilizzo in SUMO, è estremamente utile in combinazione con TraCI poichè tramite uno script scritto in Python è possibile lanciare una simulazione e poter utilizare tutte le funzioni offerte da TraCI descritte nel paragrafo precedente.
SUMO nella sua documentazione non concede alcuna guida riguardo quale linguaggio di programmazione conviene utilizzare ma la presenza di numerosi script in Python offerti stesso da SUMO rendono la scelta di quale linguaggio di programmazione utilizzare più chiara.
Poichè tutte le simulazioni hanno avuto bisogno di sfruttare TraCI, per poter elaborare la grande mole di dati generati dalla simulazione, è stato necessario sfruttare numerose librerie presenti in Python.
Per esempio sono state utilizzate *"Plotly"* è una libreria che permette di poter generare in Python dei grafici a partire da file *.csv*,oppure *CSV* è stata utilizzata per poter creare files *.csv* a partire dai file di output forniti da SUMO al termine della simulazione.

2.4 OpenStreetMap OSM

OpenStreetMap (OSM) è un progetto mondiale, libero e collaborativo per la raccolta di dati geografici [21]. OSM offre una mappa liberamente modificabile, del mondo intero, rilasciata con una licenza libera, contiene informazioni riguardo strade, sentieri, caffè, stazioni ferroviarie e molto altro ancora. OpenStreetMap, utilizzato con SUMO, permette di poter scaricare una mappa dettagliata del mondo reale e poter lanciare le diverse simulazioni modificando i parametri necessari dall'utente. Nei paragrafi precedenti è stata citata la presenza, nel pacchetto di SUMO, di numerosi script in Python che possono aiutare l'utente nella realizzazione della simulazione.
Tra questi si trova *osmWebWizard.py*, tale script offre una delle soluzioni più semplici per iniziare a lanciare simulazioni su scenari reali in SUMO. Sulla base di una selezione di un estratto della mappa di OpenStreetMmap, è possibile configurare una domanda di traffico randomizzata ed eseguire e visualizzare lo scenario generato. Dalla Figura 2.11 si può notare come sia semplice , tramite questo tool offerto da SUMO, costruire

scenari differenti. L'utilizzo di questo servizio è ottimo per poter effettuare simulazioni di prova in modo molto rapido, poichè è possibile effettuare scelte riguardo porzione della mappa, veicoli nella simulazione e intensità del traffico superficialmente. OsmWebWizard si occuperà di generare tutti i file necessari, già in *.xml*, per poter lanciare una simulazione in SUMO ed esso rappresenta uno strumento estremamente importante.

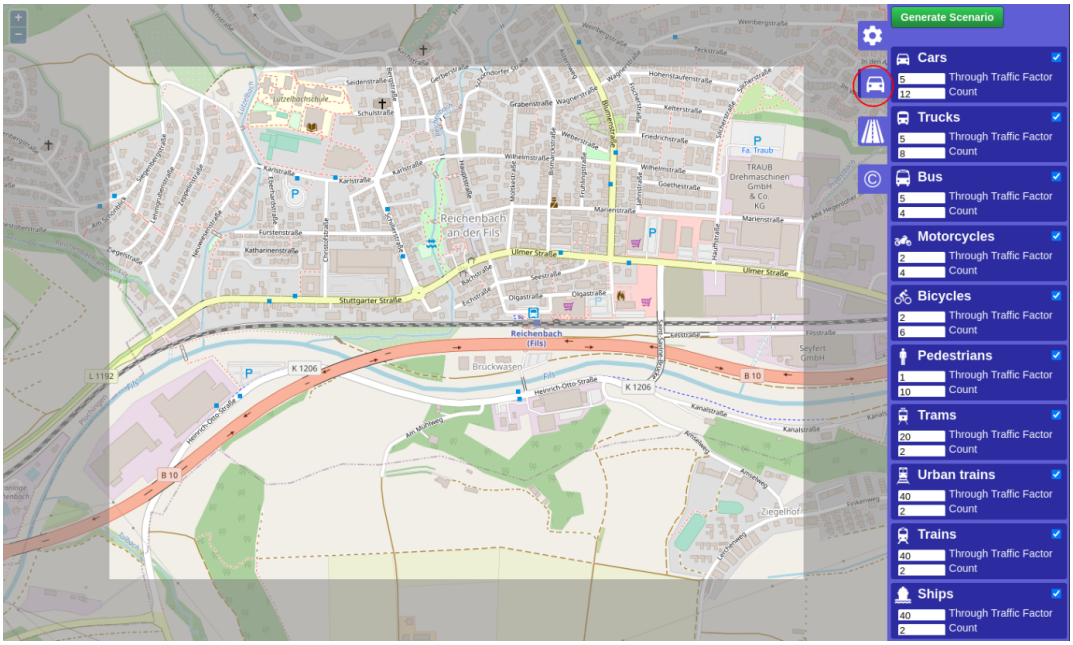


Figura 2.11: Da documentazione di SUMO [18] OSMWebWizard

Capitolo 3

Valutazione Empirica

L'obiettivo principale di questo studio è di indagare l'impatto sulla ricerca di un parcheggio in base a diverse strategie di guida sfruttando sistemi differenti di Parking Guidance, in diverse condizioni di traffico. In questo capitolo verranno descritti tutti gli step intrapresi durante l'intero studio, descrivendo i problemi incontrati e giustificando le scelte. Le simulazioni sono state eseguite utilizzando il simulatore SUMO [18], mentre i dati ottenuti sono stati elaborati mediante l'utilizzo di appositi script in Python [20].

3.1 Obiettivi e metodologia

"Vale davvero la pena avvalersi di sistemi di Parking Guidance?"

Per poter rispondere a questa domanda, in tale studio si presenta un'analisi basata su dati di parcheggio ottenuti da diverse simulazioni di scenari reali. La simulazione è caratterizzata da tre approcci che si riferiscono ai tre tipi di informazioni che un PGS fornisce. [1.2.1][1.2.2][1.2.3]
Ogni scenario è contraddistinto da situazioni altamente realistiche in cui i conducenti sono intenti a circolare per un'area urbana in cerca di un parcheggio e tramite tali simulazioni è possibile ottenere numerosi dati utili per rispondere alla domanda. Questo approccio simulerà i diversi comportamenti di un guidatore sia nel caso in cui non sia supportato da alcun sistema di Parking Guidance e sia se nel caso in cui sia totalmente informato riguardo ubicazione dei posti auto, capacità dei singoli parcheggi e disponibilità.
Terminate le simulazioni dei vari scenari, infine, si possono ottenere dati riguardo la distanza percorsa da ogni veicolo, tempo impiegato nel cercare un parcheggio, carbura-

te utilizzato o calcolare il quantitativo di CO2 emesso. Tramite questi dati è possibile stabilire effettivamente i benefici e poterli quantificare. Un ipotetico scenario realistico può essere quello in cui gli enti di viabilità del comune di una determinata città hanno intenzione di fare un investimento su tecnologie di Parking Guidance e capire, prima di impiegare un'ingente somma di denaro in questo campo, i reali benefici che può comportare tale scelta e a quale tipologia di sistemi di PGI affidarsi.
Per poter fare ciò sono state prese in analisi tre tipi di strategie..

3.1.1 Strategia non informata

Una Strategia totalmente non informata si basa sul simulare scenari in cui lungo le strade delle città non è presente alcun sistema di Parking Guidance. Seguendo questo principio, è identificata una situazione molto comune nelle grande maggioranza dei centri urbani in cui i veicoli, in cerca di parcheggio, circoleranno senza avere alcuna informazione riguardo l'ubicazione, la capacità e la distanza dalle aree di sosta.
I risultati attesi, dopo essere stati organizzati ed elaborati, se riusultassero altamente negativi darebbero un'importante risposta in merito alla questione di adottare o meno sistemi di PGS.

3.1.2 Strategia con informazione statica

Una Strategia con informazione statica simula il caso in cui siano presenti, nel centro urbano preso in analisi, sistemi di PGI con informazione statica. Sistemi di Parking Guidance sfruttati al fine di ottenere informazioni di tipo statico si basano sull'utilizzo delle mappe o su dati che raffigurano l'affluenza delle strade in un certo arco della giornata. Sfruttando tali sistemi, i guidatori sono informati riguardo il posizionamento dei parcheggi e sono in grado di poter individuare quelli più vicini alla propria destinazione. Tali tecnologie garantiscono ai conducenti di conoscere l'ubicazione dei parcheggi più grandi così da evitare il girare senza punti di riferimento un quartiere. Per poter capire se effettivamente in un determinato contesto preso in analisi questo tipo di tecnologia può bastare per ridurre gli indici di traffico nelle strade, verrà effettuata un'analisi dei dati ottenuti dalle simulazioni.

3.1.3 Strategia con informazione dinamica

Questo tipo di Strategia simula un contesto urbano in cui siano stati installati sistemi di Parking Guidance al fine di ottenere informazioni dinamiche riguardo la capacità in tempo reale dei posti auto.

I risultati ottenuti rappresentano il caso in cui in un centro urbano siano installati sensori o dispositivi in grado di poter tracciare l'entrata e l'uscita dei veicoli dai parcheggi. Questo tipo di tecnologie garantiscono ai conducenti di conoscere non solo l'ubicazione dei posti auto, ma anche se il parcheggio più vicino alla propria destinazione è libero o meno.

3.2 Dataset

Questo studio utilizza un modello microscopico del traffico stradale con dati sull'attività di viaggio locale per simulare il comportamento dei veicoli in *San Francisco Bay Area* a fine di analizzare il flusso del traffico, i chilometri percorsi o le emissioni dei veicoli. In questo dataset [22] è presente l'intero set dei dati riguardo il flusso dei veicoli nella business area di San Francisco, e tutte le informazioni necessarie riguardo la reale ubicazione dei parcheggi su strada.
La scelta del centro urbano da analizzare è ricaduta su San Francisco poichè, oltre ad essere una delle città dove si verifica maggiormente il "problema del parcheggio" [16], presenta un dataset aggiornato direttamente dal comune riguardo viaggi e parcheggi su strada.
Al fine di poter svolgere simulazioni altamente attendibili, tutti i percorsi generati sono stati ottenuti da [23], una mappa offerta direttamente dal comune di San Francisco che mostra il tragitto dei veicolo da un quartiere ad un altro nell'area urbana.

3.3 Implementazione delle strategie in SUMO

Come descritto all'inizio del capitolo, l'obiettivo è quello di simulare le tre strategie analizzate precedentemente, per poter fare ciò sono stati utilizzati i dati relativi a mappa, posizione e capienza dei parcheggi, viaggi dei veicoli, ottenuti dal dataset offerto dal comune di San Francisco [22].
Le simulazioni relative alle tre strategie hanno una durata di 24 ore (circa 86400 steps), presentano circa 13000 veicoli e 5588 parcheggi su strada.

3.3.1 Strategie su Scenari di prova

Prima di iniziare ad applicare le diverse strategie che si basano sui differenti sistemi di Parking Guidance in scenari reali, è stato necessario costruire brevi mappe demo per poter comprendere ed utilizzare al meglio il simulatore SUMO.

Con l'aiuto di NetEdit è stato possibile creare diversi scenari con differenti richieste e competitività di parcheggi al fine di poter ricreare differenti casistiche che si possono verificare in un determinato incrocio di una mappa. Svolgendo tali prove è stato possibile analizzare e risolvere due problematiche che sono state fonte di errori sopratutto nella simulazione della Strategia 1.
Il primo errore che si è riscontrato riguarda il caso in cui un parcheggio si trova all'inizio di una strada e un determinato veicolo viene reindirizzato verso eso. Poichè il reindirizzamento entra in gioco solo nel momento in cui il veicolo attraversa un dispositivo di Parking Rerouter, esso non riuscirà a frenare in tempo e costringerà SUMO a lanciare un errore della simulazione. Per poter risolvere tale problema si sono studiate diverse soluzioni, la prima è stata quella di spostare i dispositivi di Rerouting ad una distanza tale da non rendere la frenata dei veicoli troppo brusca.
Tale risoluzione del problema, però non è facilmente praticabile poichè su scenari con mappe molto grandi si dovrebbe modificare manualmente la posizione di oltre 5000 dispositivi di Rerouting. Una valida soluzione è possibile tramite l'utilizzo di TraCI, dove con uno script in Python è stato possibile effettuare il catch dell'eccezione che viene lanciata da SUMO e stabilire un nuovo parcheggio in cui reindirizzare il veicolo, così che non dovrà effettuare alcuna frenata di emergenza per poter provare a parcheggiare nel parcheggio presente in quella strada.

Il secondo problema che è stato riscontrato nasce nel caso in cui due veicoli vengono reindirizzati verso lo stesso parcheggio con capacità 1 nel momento in cui si trovano entrambi sulla stessa strada. Questo caso molto raro si verifica poichè entrambi i veicoli, nel momento in cui vengono reindirizzati verso una nuova area di parcheggio dal Parking Rerouting, non sono in grado di poter prevedere che tale parcheggio è stato già individuato da un altro veicolo.
Il comportamento, errato dei veicoli è che il primo veicolo si parcheggia nell'unico posto disponibile, mentre il secondo resta in attesa che esso venga liberato occupando la lane. Il risultato che si vuole ottenere è che il secondo veicolo, una volta che ha verificato che il parcheggio non è più disponibile, possa continuare il proprio tragitto verso nuove aree di sosta.
Per risolvere tale problema è stato utile nuovamente TraCI, poichè tramite un algoritmo, i veicoli nel momento in cui entrano in fase di attesa che un parcheggio si liberi per oltre n steps, verranno automaticamente reindirizzati verso una nuova area di parcheggio. Quindi il secondo veicolo in questo caso, una volta atteso un paio di secondi, continuerà la propria ricerca in altre strade.

3.3.2 Elaborazione del dataset

Per poter effettuare simulazioni attendibili si è suddivisa la mappa di San Francisco in quartieri, tale suddivisione è stata fatta seguendo i dati di viaggio tra quartieri ottenuti da [23]. Il dataset offre un file denominato *san-francisco.poly*, dove è presente la suddivisione per quartieri dell'intera area. Tramite uno script in Python offerto da SUMO, *edgesInDistricts*, è possibile inserire come paramtri la mappa (*.net*)e il file (*.poly*) per generare un file *.taz* contenente tutti i distretti della mappa.
Questo file *.taz*, a differenza di *san-francisco.poly* effettua una suddivisione in quartieri non dal punto di vista grafico bensì assegnando ad ogni strada della mappa un diverso distretto. In questo modo è possibile creare percorsi da un distretto ad un altro utilizzando le singole strade come punto di partenza e punto di arrivo. Tutto questo è necessario poichè SUMO, per poter costruire un nuovo percorso, accetta come parametri di punto di partenza e destinazione esclusivamente strade (*edges*).

```
1        <additional>
2        <poly id="1" color="blue" fill="true" layer="0"
3        shape="3077.42414,3008.42452..." \>
4        <\additional>
5
```

Listing 3.1: Esempio di file.poly

```
1        <tazs>
2        <taz id="1" color="blue" edges="-254466686#0 -254466686#1 ..." \>
3        <\tazs>
4
```

Listing 3.2: Esempio di file.taz

Da questi due esempi si può notare come i due files si differenziano solo dall'ultimo parametro, dove il primo si riferisce alla forma del distretto, mentre il secondo elenca tutte le *edges* che compongono quel distretto.
Il risultato ottenuto è visibile anche graficamente nella gui di SUMO come in Figura 3.1.
La suddivisione della mappa in quartieri è stata necessaria per poter risolvere una problematica che è stata riscontrata durante tutte le simulazioni. I veicoli, seguendo le differenti strategie potevano individuare come prossimi parcheggi da visitare anche aree di sosta molto distanti dalla propria destinazione finale.
Essa rappresenta una casistica estremamente lontana dalla realtà, un conducente che necessita di svolgere un servizio in una determinata strada è impensabile che possa parcheggiare a chilometri di distanza da essa. Per poter risolvere tale problema, quindi

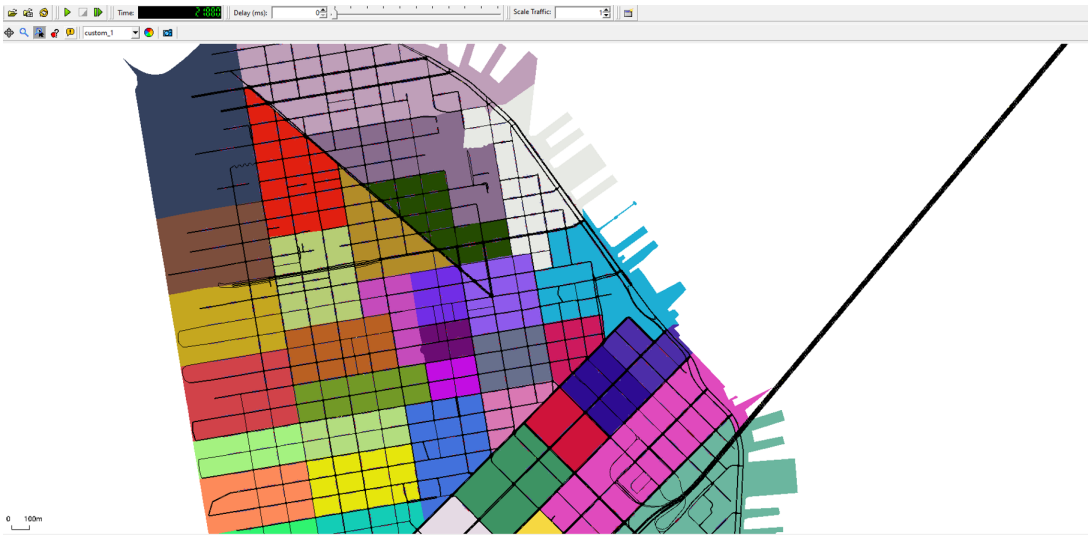


Figura 3.1: Mappa SanFrancisco suddivisa in quartieri

è stato necessario suddividere la mappa in quartieri realmente esistenti, in modo che i veicoli possano cercare un parcheggio solo se si trovano nel proprio quartiere di destinazione.
Questo è stato possibile creando differenti sistemi di Parking Area Rerouter per ogni quartiere. Si verifica quindi che, un veicolo una volta entrato nel proprio quartiere di destinazione e visitato il fake parking associato ad esso, attiverà il rerouter corrispondente a quel determinato distretto e di conseguenza il conducente verrà reindirizzato nei soli parcheggi presenti nella sua area di destinazione.

3.3.3 Creazione di Veicoli e Tragitti

Per la creazione dei veicoli e dei relativi tragitti si è utilizzato il file *trip.add.xml* presente nel dataset di partenza [22]. Questo file è un'elenco di tutti i veicoli che verranno generati nel corso della simulazione, presentano parametri relativi all'istante in cui verrà generato un veicolo, la categoria a cui appartiene e la *lane* di partenza o destinazione. Ad ogni viaggio è possibile assegnare un parcheggio al quale o il veicolo deve necessariamente sostare prima di concludere il proprio viaggio. Nel file originale [22] non sono presenti tali soste, di conseguenza è stato necessario inserirle.
Svolgere simulazioni su scenari molto ampli come può essere quello di un'intera città, rende diffcile la modifica o l'aggiunta di campi per ogni veicolo, basta pensare che nelle simulazioni svolte in questo studio sono presenti circa 13000 veicoli e apportare modifiche "a mano" è un lavoro pressocchè impossibile. Per superare tale problema

è stato necessario scrivere diversi script Python a fine di modificare ricorsivamente gli opportuni file *.xml*.

```
1        <trip id="3770" color="1,1,0" from="142972028#1" to="283537380#0" depart="6">
2        <stop duration="1200" key="parking.capacity.weight" value="500"
3        parkingArea="fakeparking_8" />
4        </trip>
5
```

Listing 3.3: Esempio di trip file

Il Codice 3.3 rappresenta un frammento del trip file, esso rappresenta il comportamento del veicolo con *id*=3770 che partirà da una lane specifica di un determinato quartiere e arriverà in un'altra lane. La parte relativa all'attributo *stop* riguarda il parcheggio in cui il veicolo dovrà sostare, seguendo quale algoritmo e per quale durata.
Il trip file, infine rappresenta il cuore della simulazione, la parte in cui verrà gestito il comportamento di ogni veicolo e che sarà differente a seconda della strategia analizzata.

3.3.4 Il Rerouting e i FakeParking

Prendendo in esame il frammento di Codice 3.3 che rappresenta il tragitto che il veicolo 3770 dovrà svolgere, il parametro *key* rappresenta il tipo di algoritmo di Rerouting da utilizzare. Come descritto in 2.1.4 il Rerouting è un dispositivo che permette di reindirizzare un veicolo verso una nuova destinazione o parcheggio. Sono stati presentati in Figura 2.9 l'elenco di tutti gli algoritmi di ParkingRerouting presenti in SUMO. A seconda dell'algoritmo scelto è possibile dare un criterio riguardo la decisione del prossimo parcheggio che un veicolo visiterà. Tali algoritmi offerti da SUMO si attiveranno solo nel momento in cui il veicolo, una volta incontrato il primo parcheggio con capacità uguale a 0, inizierà la propria ricerca.
Per fare in modo che l'algoritmo scelto verrà attivato per ogni veicolo sarà necessario che il primo tentativo di parcheggio fallisca, poichè altrimenti verrà selezionato quello impostato nel campo *parkingArea* del trip file. Per questo motivo sono stati inseriti all'interno della mappa 40 parcheggi "finti", uno per ogni quartiere. I parcheggi "finti" (chiamati *fake parking*) hanno capienza uguale a 0 e dimensione nulla, in modo tale da risultare graficamente invisibili. Di seguito è rappresentato un esempio del comportamento dei veicoli in uno scenario semplificato.

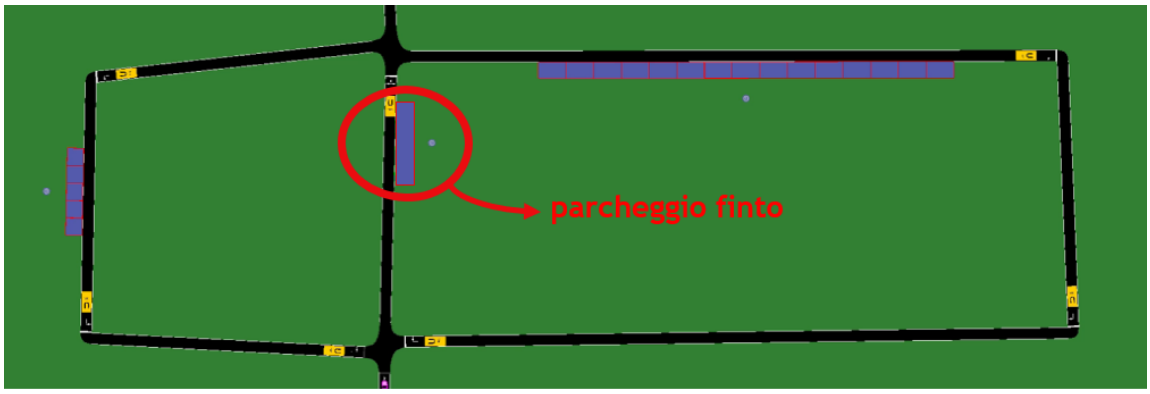


Figura 3.2: Mappa Esempio

In Figura 3.2 è rappresentata la mappa, costruita in NetEdit [2.1.1], dove il parcheggio cerchiato è quello fittizio (con capienza = 0). Nella mappa sono rappesentati anche i dispositivi di Rerouting (rettangoli gialli) che sono necessari per il reindirizzamento dei veicoli.

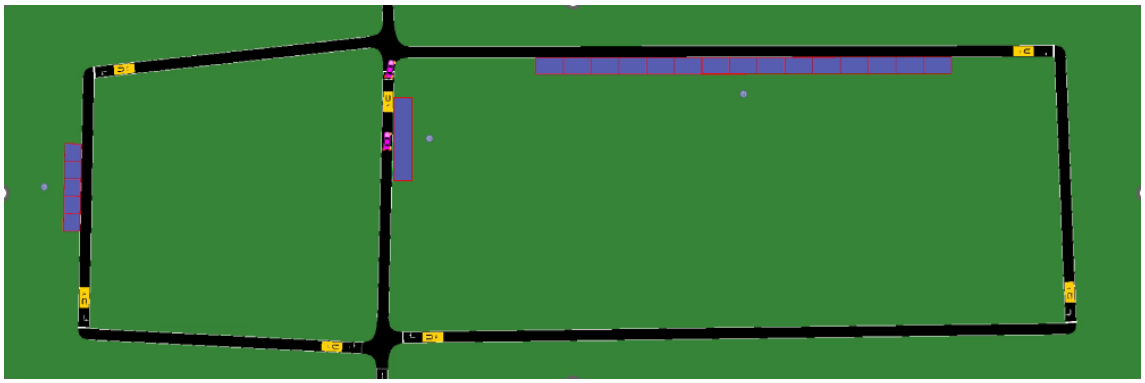


Figura 3.3: Esempio Step 1

Mandando avanti la simulazione, vengono generati 2 veicoli che hanno nell'apposito trip file, il campo *parkingArea* impostato al parcheggio finto. Si può notare come i due veicoli, poichè il proprio parcheggio di destinazione non ha posti liberi, non si fermeranno in prossimità del parcheggio fittizio, bensì continueranno la propria ricerca [Figura 3.3].
Nell'istante in cui falliscono il primo parcheggio si attiverà l'algoritmo di Rerouting impostato come paraemtro dall'elenco in Figura 2.9.

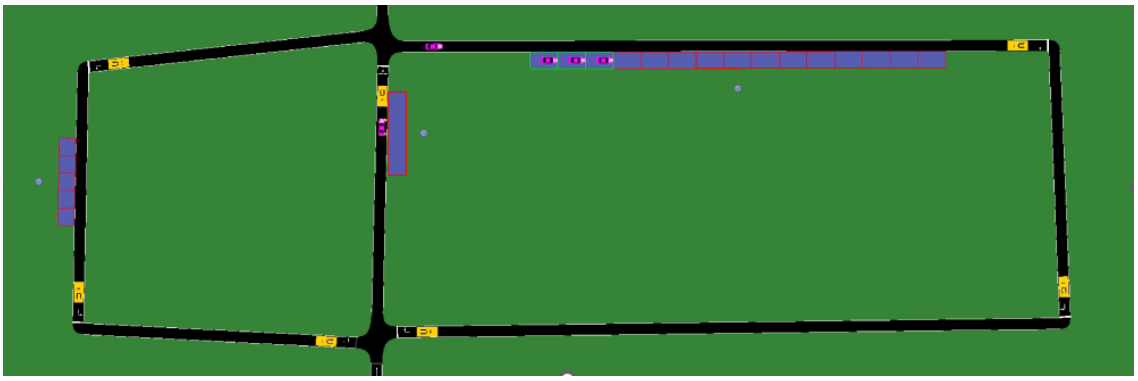


Figura 3.4: Esempio Step 2

Continuando la simulazione, verranno generati anche altri veicoli che, ignorato il parcheggio fittizio, andranno a parcheggiare nell'area di sosta che l'algoritmo scelto determinerà [Figura 3.4].
Di seguito verranno presentate le tre Strategie intraprese, già teoricamente in 3.1, analizzando come sono state realizzate e mostrando le soluzioni alle problematiche riscontrate durante la realizzazione.

3.3.5 Strategia 1

Questa strategia, definita come non informata, simula uno scenario in cui l'intera città di San Francisco non dispone di alcun sistema di Parking Guidance, di conseguenza i conducenti non conoscono alcuna informazione riguardo i parcheggi. Nello specifico i veicoli non conoscono, dimensione, disponibilità e ubicazione dei parcheggi.
Sfruttando le nozioni di ParkingRerouting e degli algoritmi già definiti in SUMO, dalla Figura 2.9 è stato scelto, almeno inizialmente di applicare l'algoritmo *parking.probability.weight*. Tale algoritmo è stato scelto poichè i veicoli avrebbero cercato un prossimo parcheggio in maniera del tutto casuale dall'elenco di tutti i parcheggi. La scelta di questo algoritmo può sembrare ottimale, poichè i veicoli non conoscono la disponibilità dei singoli parcheggi e quindi verranno reindirizzati verso altre aree di sosta casualmente.
Tale approccio non è del tutto corretto poichè con questo algoritmo i veicoli saranno a conoscenza della posizione dei parcheggi. Il risultato da ottenere ad un incrocio è che i veicoli sceglieranno in quale strada svoltare casualmente, senza sapere se effettivamente su quella strada è presente un parcheggio.

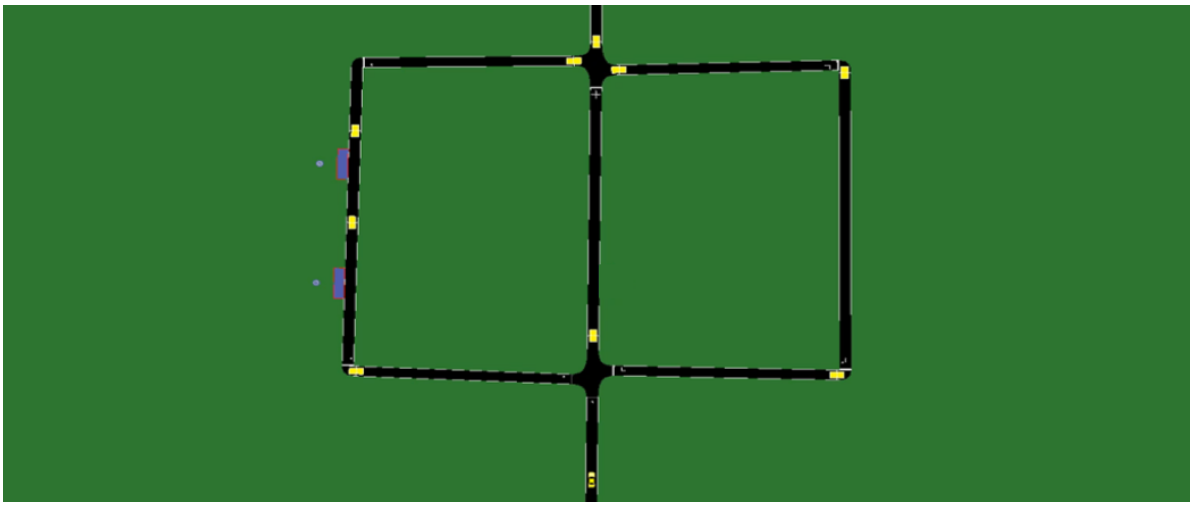


Figura 3.5: Esempio Strategia 1

Per esempio, in Figura 3.5 il risultato atteso è che i veicoli svolteranno casualmente a destra o a sinistra, mentre nel caso in cui verrà sfruttato l'algoritmo *parking.probability.weight* i veicoli, poichè verranno reindirizzati nel primo o nel secondo parcheggio, svolteranno tutti a sinistra.

La soluzione per risolvere questa problematica è stata quella di costruire un algoritmo di Rerouting verso aree di parcheggio da zero, utilizzando TraCI [2.2]. Tale algoritmo, invece di reindirizzare i veicoli verso nuovi parcheggi, funziona in modo tale da assegnare ad ogni incrocio una nuova strada di destinazione e, nel caso in cui durante il proprio tragitto incontrerà un parcheggio libero potrà parcheggiare. Una volta che la propria sosta è terminata il veicolo verrà reindirizzato verso la sua strada di destinazione originale.
Per poter realizare questo algoritmo è stato necessario utilizzare funzioni presenti nella libreria di TraCI e dei Detector. Di seguito è presentato uno pseudocodice in cui è descritto l'algoritmo.

```
1        for (veicolo in tutti_i_veicoli_su_detector)
2            if (veicolo.incontra_parcheggio_libero and
3                veicolo.cerca_parcheggio and
4                strada.distretto==veicolo.distretto)
5                veicolo.parcheggia
6                veicolo.vai_a_destinazione
7            else
8                veicolo.cambia_destinazione
9
```

Listing 3.4: Algoritmo di Rerouting Strategia 1

Alla linea 1 del codice verranno presi in analisi, ad ogni step, tutti i veicoli che si trovanosu un detector. Se il veicolo non ha incontrato un parcheggio libero, dovrà cercarlo in un'altra strada, altrimenti potrà parcheggiare. Al termine della propria sosta proseguirà il proprio tragitto, senza fermarsi ad altri parcheggi, verso la sua destinazione originaria.
In linea 4 è presente una verifica del distretto poichè, per evitare che un veicolo possa cercare parcheggio in una strada troppo distante dal suo quartiere di destinazione, la ricerca è limitata al solo distretto in cui deve arrivare.
Il comportamento di un veicolo sarà quindi quello in cui arrivato ad un incrocio verrà reindirizzato verso una nuova strada, e non verso un nuovo parcheggio.

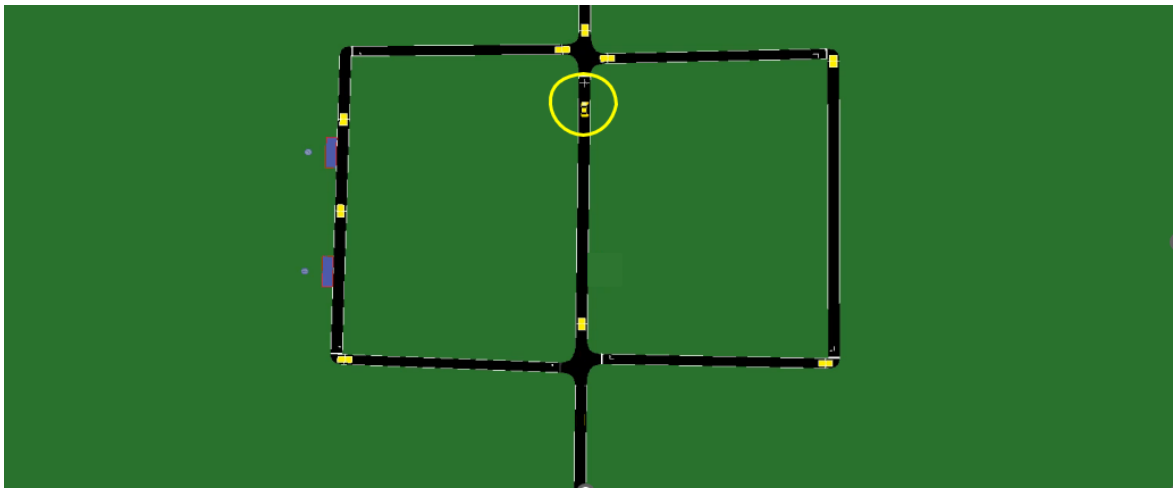


Figura 3.6: Esempio Strategia 1 con algoritmo in TraCI

In Figura 3.6 si può notare un veicolo ad un incrocio, esso non conosce la reale posizione dei parcheggi, quindi in uno scenario realistico, la scelta del guidatore non verrà influenzata in alcun modo dalla posizione delle aree di sosta. Per tale motivo il veicolo potrà svoltare a sinistra o a destra.

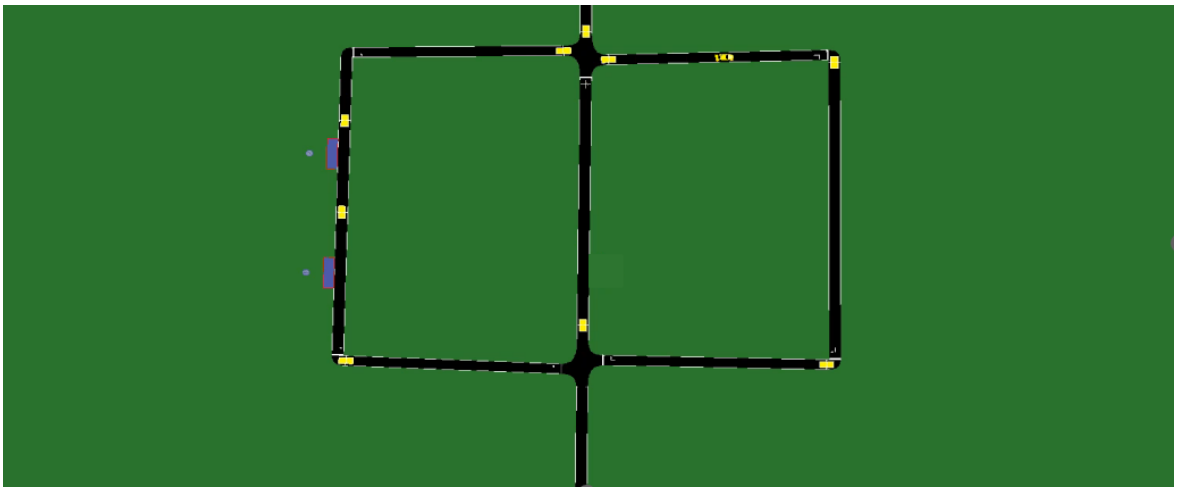


Figura 3.7: Esempio Strategia 1 con algoritmo in TraCI

Nell' esempio in Figura 3.7 il veicolo infatti svolterà a destra effettuando quindi una scelta che non lo porterà ad alcun parcheggio, poichè il conducente non sarà a conoscenza della disponibilità, della dimensione e della posizione del parcheggio sulla mappa.
In questa strategia quindi, ad ogni incrocio, il veicolo effettua una scelta casuale sulla prossima strada da prendere senza allontanarsi troppo dalla sua destinazione originale.

3.3.6 Strategia 2

Questa seconda strategia simula il comportamento dei veicoli in una mappa con Parking Guidance ad informazione statica. I veicoli hanno informazione riguardo dimensione e posizione dei parcheggi.L'idea di base è quella in cui un veicolo una volta che è entrato nel suo quartiere di destinazione inizierà la sua ricerca di parcheggio.
La scelta del parcheggio da visitare è vincolata dalla capacità del parcheggio stesso, ovvero i conducenti guideranno verso l'area di sosta su strada più grande. Facendo così i veicoli avranno più possibilità di trovare parcheggio. Per fare ciò si è utilizzato l'algoritmo di SUMO *parking.capacity.weight*, che segue l'idea in cui la capacità dell'area di parcheggio è elemento vincolante per la scelta dove parcheggiare. In questa strategia i veicoli giunti ad un incrocio, non effettueranno più la propria scelta individuando la prossima strada da visitare, bensì sarà l'algoritmo di Rerouting a individuare il parcheggio con la capacità maggiore e a indirizzare il veicolo verso esso. Per poter verificare il comportamento dei veicoli in questo algoritmo di seguito è presentata una dimostrazione in uno scenario costruito su NetEdit.

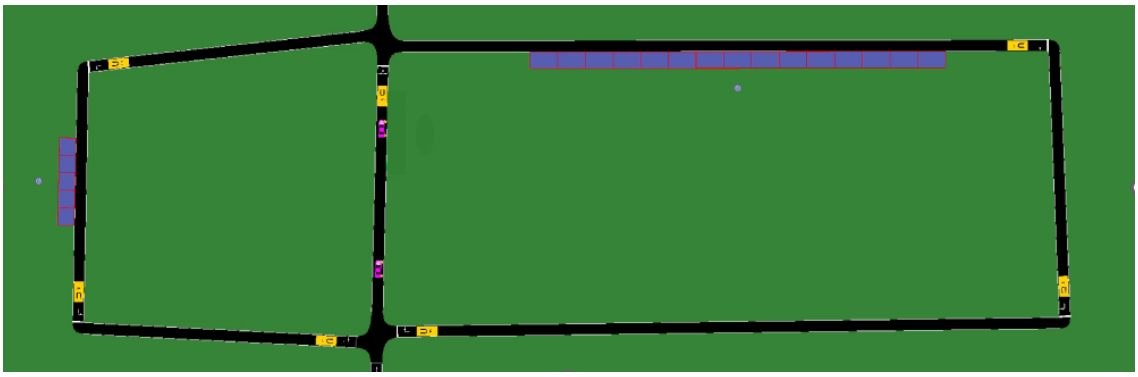


Figura 3.8: Esempio Strategia 2 - Step 1

In Figura 3.8 è rappresentato lo scenario preso in esempio, in esso sono presenti due veicoli che stanno per intraprendere un incrocio e due parcheggi di capienza differente. Il parcheggio che si trova alla destra dell'incrocio è più capiente e in questa strategia i veicoli, una volta attraversato il dispositivo di Rerouter, andranno a parcheggiare verso questo parcheggio.

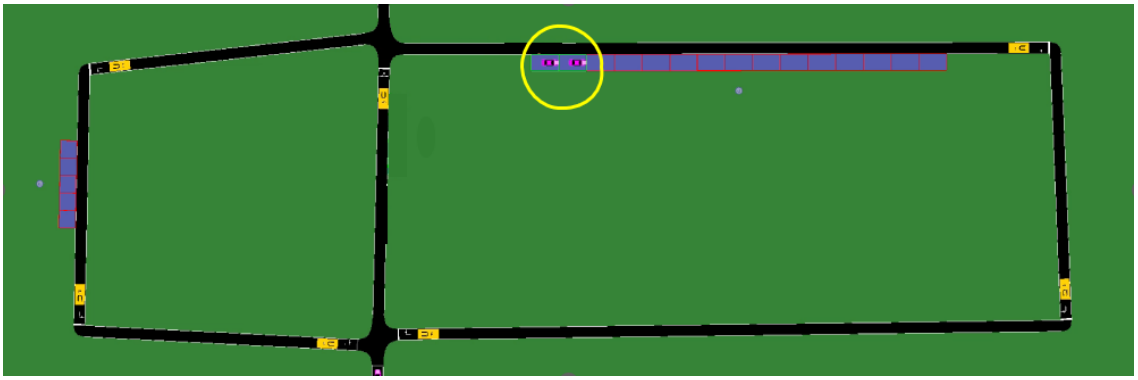


Figura 3.9: Esempio Strategia 2 - Step 2

Andando avanti con la simulazione i veicoli verranno reindirizzati dal Rerouter verso il parcheggio più grande, di conseguenza giusnti all'incrocio svolteranno a destra [Figura 3.9].

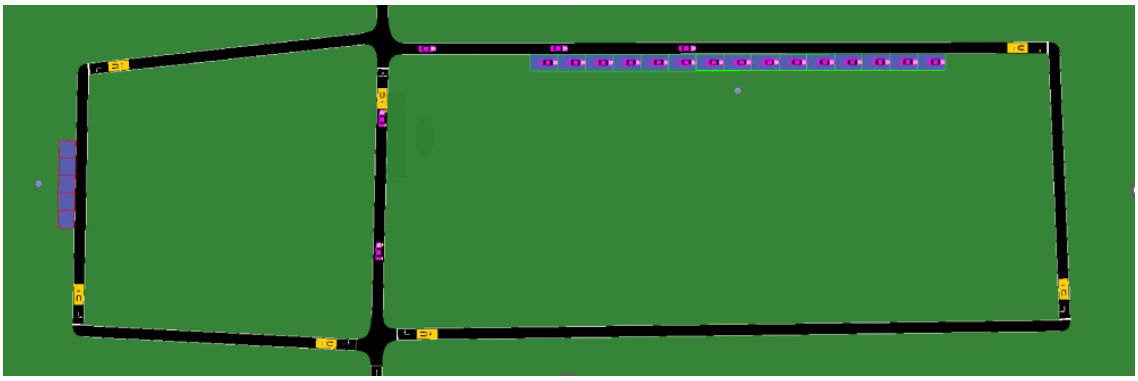


Figura 3.10: Esempio Strategia 2 - Step 3

Quando il parcheggio più capiente sarà saturo, gli altri veicoli continueranno ad essere reindirizzati verso questo parcheggio poichè, non essendo muniti di dispositivi di Parking Guidance dinamici, non conosceranno in tempo reale la disponibilità delle aree di sosta. Per questo motivo nell'esempio in Figura 3.10 i veicoli continueranno a svoltare a destra all'incrocio, e una volta che non riusciranno a completare il parcheggio verranno reindirizzati verso l'altro.

3.3.7 Strategia 3

L'ultima strategia rappresenta un upper bound alle strategie che è possibile realizzare sfruttando modelli predittivi. Rappresenta una strategia con informazione dinamica, ovvero uno scenario in cui il guidatore conosce la disponibilità di posti in tempo reale. I veicoli sono a conoscenza della dimensione, ubicazione, capienza e disponibilità in tempo reale dei parcheggi. Con questa strategia i conducenti, conoscendo l'esatta posizione dei parcheggi e conoscendo anche la loro disponibilità in tempo reale non falliranno mai il proprio parcheggio. L'unico caso in cui un veicolo possa fallire il proprio parcheggio si verifica quando un conducente individua il parcheggio da visitare e mentre tenta di raggiungerlo tale area di sosta viene completamente occupata.
Questa strategia in SUMO è realizzabile utilizzando l'algoritmo di Rerouting *parking.absfreespace.weight* che basa la scelta dei parcheggi in base alla disponibilità. Per poter rendere nota la capacità di un parcheggio in tempo reale è necessario che la visibilità di esso sia settata a *true* così come descritto al Paragrafo 2.1.4.

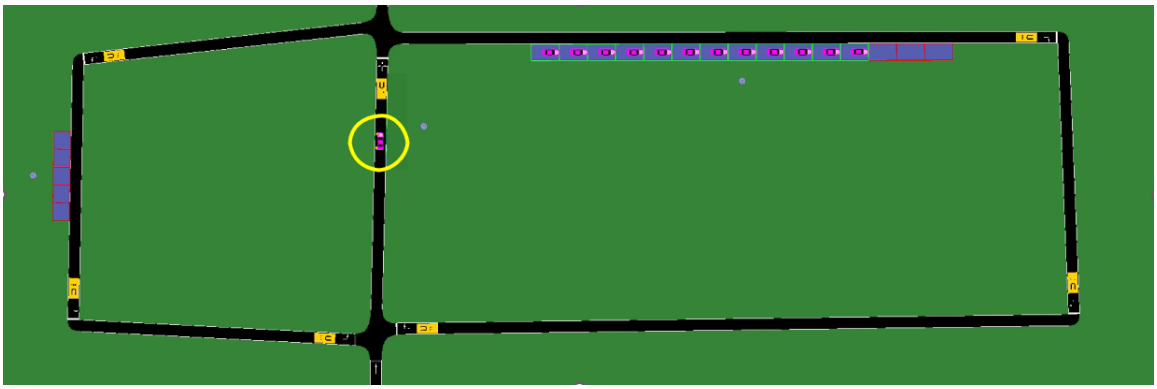


Figura 3.11: Esempio Strategia 3 - Step 1

In Figura 3.11 è rappresentato un esempio di un caso particolare in cui sono presenti due parcheggi, uno con disponibilità = 3 e un altro con disponibilità = 5. Dopo che tutti i veicoli si sono recati verso il parcheggio con capacità e disponibilità maggiore, il veicolo preso in esame in questa strategia i recherà verso il parcheggio con più posti liberi come in Figura 3.12

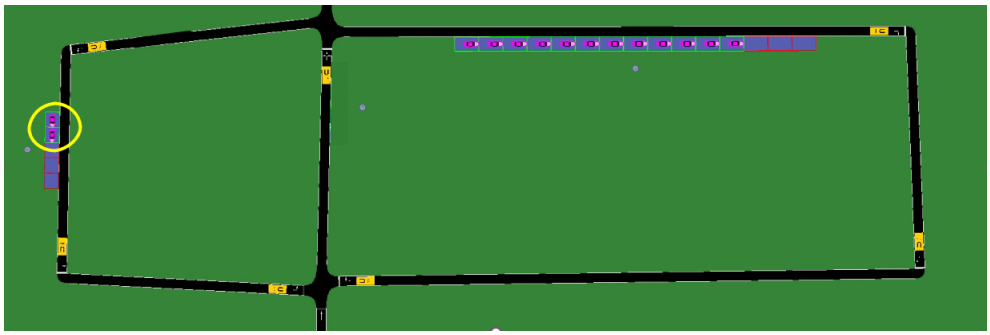


Figura 3.12: Esempio Strategia 3 - Step 2

3.4 Elaborazione dei dati ottenuti

Nei paragrafo precedente è stata descritta la simulazione di tre scenari differenti al fine di poter analizzare i benefici che può portare l'adozione di sistemi di Parking Guidance. Per poter generare i dati ottenuti, SUMO offre la possibilità di poter creare, terminata una simulazione, dei file di output come descritto nel Paragrafo 2.2.1. I file di output generati sono in particolare, *vehroutes.xml* e *tripinfo.xml*. Per quanto riguarda il primo file, è possibile ottenere l'intero percorso e i parcheggi visitati da un veicolo durante la simulazione. Questo file è legato al comportamento di ogni singolo veicolo durante la simulazione, definendo anche il tempo di durata del parcheggio e l'istante in cui è stato reindirizzato in una nuova area di sosta.

```
        <vehicle id="3770" depart="6.00" color="yellow" arrival="1573.00">
        <routeDistribution>
            <route replacedOnEdge="-202485361#1" reason="8:parkingAreaReroute"
            replacedAtTime="328.00" probability="0" edges="1429720#1 896695#0 ..."/>
        </routeDistribution>
        <stop parkingArea="parkingArea_-202485361#1_0_445" duration="1200.00"/>
    </vehicle>

```

Listing 3.5: vehroutes.xml file

Il frammento di Codice 3.5 rappresenta come sono rappresentati i veicoli in questo file, un parametro molto utile è *replacedAtTime* che identifica l'istante in cui si attiva l'algoritmo di Rerouting e quindi è il momento in cui il veicolo inizia a cercare un parcheggio. Per poter calcolare il tempo di ricerca di ogni conducente finchè non arriva a destinazione si svolge il calcolo *arrival* - *replacedAtTime* - *duration*. Tale calcolo rappresenta un dato confrontabile tra tutte le simulazioni per capire, escludendo la durata del parcheggio, il tempo necessario per ogni veicolo per cercare un posto auto e poi tornare a destinazione. Il tempo necessario per giugnere a destinazione dopo aver parcheggiato, è un ottimo indice per determinare la distanza dal parcheggio al punto di destinazione.

```
        <tripinfo id="10512" depart="18060.00" departLane="8916792_0" departPos="5.10"
         arrival="18157.00" arrivalLane="247966218#3_0" arrivalPos="66.94"
         arrivalSpeed="11.66" duration="97.00" routeLength="1077.61"
         waitingTime="0.00" waitingCount="0" stopTime="0.00" timeLoss="7.68"/>

```

Listing 3.6: tripinfo.xml file

Il file *tripinfo.xml* contiene i dati specifici di ogni veicolo, per esempio distanza percorsa, tempo impiegato, CO2 emesso, velocità media o altri parametri. Essi sono fondamentali sopratutto per poter fare una stima in termini ambientali riguardo il CO2 emesso dai

singoli veicoli durante la simulazione.
Una volta che sono stati generati i due file descritti precedentemente, bisogna elaborarli per poter effettuare un confronto effettivo tra le varie strategie. Per poter organizzare i dati ottenuti è stato necessario alla fine di una simulazione, tramite TraCI, indicizzare i valori all'interno di un file *.csv*. Questo è stato possibile sfruttando una libreria di Python chiamata *csv*, successivamente, i dati sono stati elaborati e rappresentati graficamente con una nuova libreria, *plotly*. Lo script necessario per poter costruire i file *.csv* è stato creato da zero in Python e prende il nome di *generatecsv.py*.

3.5 Strategie con disponibilità variabile

Per poter quantificare i benefici ottenuti dalle varie strategie bisogna considerare tutti i fattori che possono incidere sul risultato finale. Tra questi è presente la disponibilità dei posti auto nell'istante in cui viene lanciata la simulazione, ovvero considereare il caso in cui i parcheggi presenti nella mappa siano parzialmente occupati. Per poter effettuare un'analisi anche in base a disponibilità dei parcheggi differenti, è stato creato uno script *setcapacity.py* dove è possibile ridurre in percentuale la capacità di tutti i parcheggi della mappa. In questo modo si possono effettuare ed analizzare simulazioni differenti in cui la competizione per parcheggiare aumenta in percentuale. Sono stati analizzati tre tipi di scenari per ogni strategia, disponibilità al 100%, disponibilità 80% e disponibilità 50%.

```
1         file = minidom.parse("parking.add.xml")
2         trip = file.getElementsByTagName('parkingArea')
3         for data in trip:
4             val = (int)(data.attributes[(str)("roadsideCapacity")].value)
5             newval = (int)((val/100)*80)
6             data.attributes["roadsideCapacity"].value = (str)(newval)
7         print(file.toxml())
8         with open("parking.add.xml","w") as f:
9             file.writexml(f)
10
```

Listing 3.7: setcapacity.py

Nel frammento di Codice 3.7 è rappresentato lo script che permette di impostare in percentuale la capacità dei parcheggi. Le linee 1 e 2 aprono il file *.xml* dei parcheggi, mentre alla linea 5 viene impostato il campo che identifica la capacità del parcheggio *roadsideCapacity* al nuovo valore. In questo caso viene impostata la capacità di ogni parcheggio all'80% della capaicità originale. Poter effettuare simulazioni su scenari

differenti in base alla capacità dei parcheggi è fondamentale per comprendere al cento per cento la bontà dei risultati che si possono ottenere. A seconda della mappa su cui sono state effettuate le simulazioni, i risultati possono essere differenti, per tale motivo è stato necessario generare diverse simulazioni modificando volta per volta i parametri relativi alla capacità dei parcheggi. Per quanto riguarda i parcheggi con capacità estremamente ridotte, per evitare che la propria capacità venga arrotondata a 0 rendendo quindi inutile il parcheggio, è stato necesssario effettuare un nuovo calcolo di tali aree di sosta. Quindi i parcheggi con capacità molto bassa hanno una probabilità impostata dall'utente di poter trasformarsi un un parcheggio vuoto. Ttutto questo è stato fatto al fine di poter creare un numero molto elevato di simulazioni, ed ottenere risultati finali ancor più affidabili.

Capitolo 4

Risultati

Nei Capitoli precedenti è stata presentata una panoramica del problema da affrontare, gli strumenti utilizzati per risolverlo e la strategia intrapresa. Lo strumento più importante è stato sicuramente il simulatore SUMO, con esso è stato possibilie generare numerose simulazioni di scenari differentei con diverse strategie di parcheggio. La domanda fondamentale a cui si è dovuto rispondere è se adottare sistemi di Parking Guidance corrisponde effettivamente ad un miglioramento in termini di ricerca di un posto auto, e quantificare i reali benefici che possono portare. Al fine di poter generare dei dati utili bisogna scegliere la tipologia più significativa, ovvero il tipo di dati necessari per confrontare le varie strategie.
Come dato principale è stato scelto quello relativo alla distanza percorsa da ogni veicolo chiamato *routeLength*. Poichè nelle tre strategie i veicoli svolgereanno gli stessi percorsi a meno della fase di ricerca del parcheggio, la differenza tra le tre casistiche sarà determinata da questa fase presa in analisi. Prima di visualizzare i dati è possibile immaginare una stima quantitativà di ciò che andremo a vedere.
I risultati attesi sono quelli di vedere effettivamente un miglioramento crescente dalla Strategia 1 alla Strategia 3 ma il quesito più importante è se il miglioramento sarà tale da poter investire in determinate tecnologie. Di seguito è presente una tabella con i dati ottenuti in base al tempo impiegato dai veicoli per cercare un parcheggio. Le classi di dati scelte sono relative a tempo di ricerca in meno di 3 minuti, tra i 3 e gli 8 minuti e più di 8 minuti.

	Meno di 3 minuti	Tra 3 e 8 minuti	Più di 8 minuti
Strategia 1	47.4%	44.1%	8.49%
Strategia 2	49.8%	49.8%	0.5%
Strategia 3	62.8%	36.4%	0.74%

Tabella 4.1: Tempo di Ricerca

La Tabella 4.1 ci mostra i primi dati sperimentali ottenuti, la Strategia 1 rappresenta che in San Francisco in uno scenario di disponimilità totale al 100% quasi la metà dei conducenti impiegano meno di 3 minuti a parcheggiare. Il dato più evidente è che circa l'8% dei veicoli impiega più di 8 minuti, a confronto con le altre due strategie è possibile notare che esiste un miglioramento notevole. Il dato più interessante è che con la Strategia 3 circa un 15% di veicoli in più impiegherà meno di 3 minuti a parcheggiare, basta questo per concludere che nello scenario di San Francisco l'utilizzo di sistemi di Parking Guidance porterebbero un netto miglioramento nei dati relativi al tempo di ricerca necessario per trovare un posto libero in cui parcheggiare da parte dei conducenti. Per poter effettuare un'analisi più approfondita è possibile analizzare anche i risultati ottenuti dalle simulazioni con disponibilità dei parcheggi differenti. Di seguito verrà presentato un confronto tra le tre Strategie con la disponibilità dei posti auto fissata al 50% della capienza, utilizzando come parametro la distanza percorsa dai veicoli.

	Meno di 5 km	Più di 5 km
Strategia 1	78.1%	21.9%
Strategia 2	82.4%	16.6%
Strategia 3	84.2%	15.8%

Tabella 4.2: Distanza percorsa con 100% di capienza

	Meno di 5 km	Più di 5 km
Strategia 1	71%	29%
Strategia 2	70.5%	29.5%
Strategia 3	67.1%	32.9%

Tabella 4.3: Distanza percorsa con 50% di capienza

In queto caso nella Tabella 4.2 si può evincere un'informazione molto importante per quanto riguarda lo scenario di San Francisco e che potrebbe essere esteso in uno scenario generico. Al diminuire della capienza dei parcheggi si può riscontrare un aumento dei veicoli che impiegano più di 5 km per svolgere il proprio tragitto, ciò vuol dire che la lunghezza del percorso dei singoli veicoli è direttamente propozionale all'aumentare della competitività della ricerca dei parcheggi.
I dati presenti nelle tabelle precedenti sono stati rappresentati anche graficamente utilizzando la libreria Python *plotly*. Con l'utilizzo di grafici è possibile scendere nel dettaglio per quanto riguarda il numero specifico dei veicoli nei vari intervalli stabiliti.

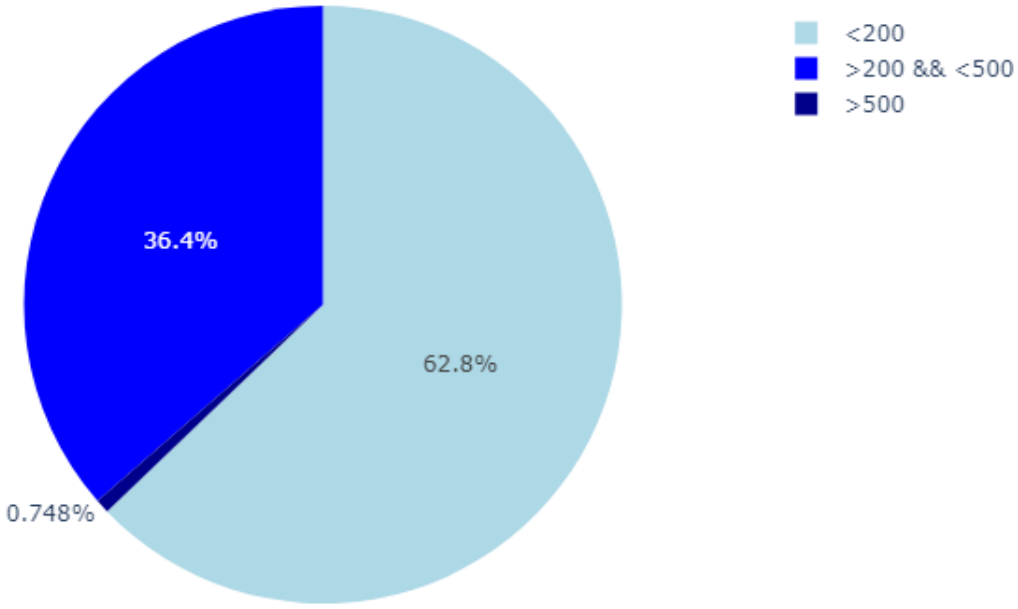


Figura 4.1: Diagramma a torta Strategia 3

In Figura 4.1 sono raffigurati i dati presenti in Tabella 4.1 della Strategia 3, l'utilizzo dei grafici è importante per poter far evincere le differenze sostanziali tra le diverse strategie. Un altro modo per rappresentare i dati ottenuti è l'istogramma. Per poter costruire un grafico di queesto tipo è necessario individuare le classi dei valori, che corrispondono alle varie fasce di tempo necessarie per trovare un parcheggio.

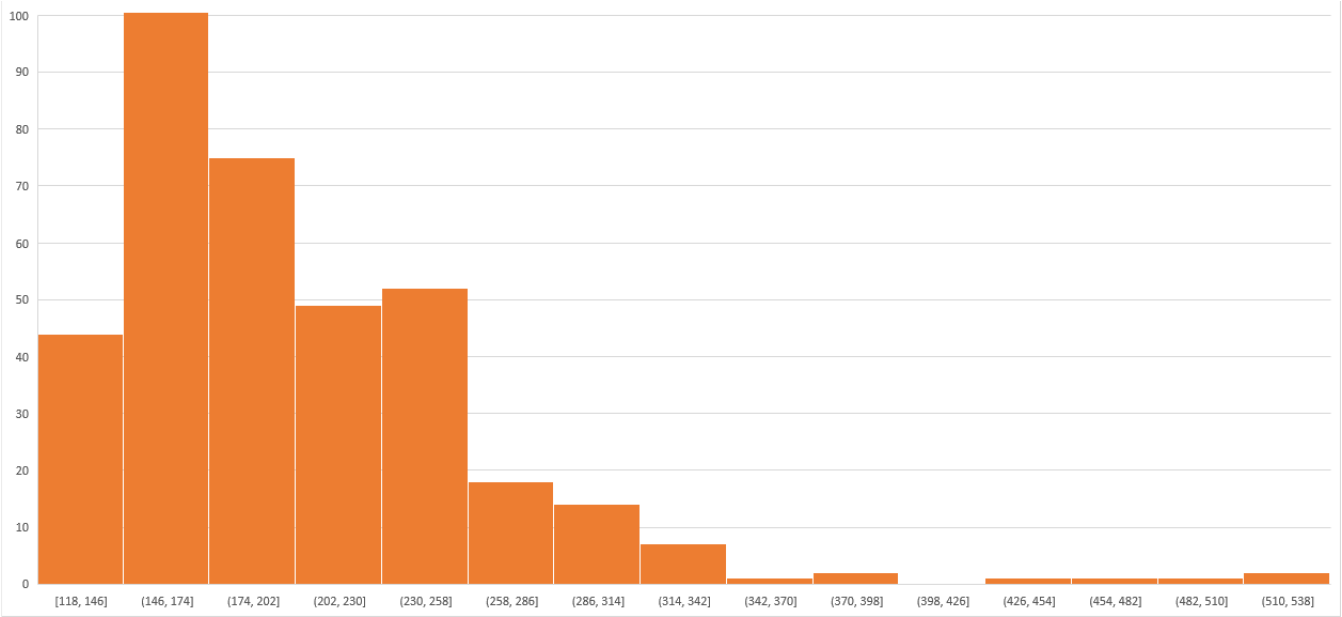


Figura 4.2: Istogramma Strategia 3

In Figura 4.2 sono rappresentati sottoforma di istogramma i dati presenti in Figura 4.1. In un istogramma la base dei vari rettangolini corrisponde alla classe di valori che è stata scelta, mentre l'area è il totale dei secondi trascorsi in ricerca di tutti i veicoli compresi nella classe. Da questo grafico si può evincere come la maggior parte dei veicoli impiegherà meno di 286 secondi per trovare un parcheggio, e la classe più numerosa è quella compresa tra 146 secondi e 174 secondi. Per poter confrontare le tre Strategie definitivamente è stato utile costruire un *box plot*. Il box plot, o diagramma a scatola e baffi viene rappresentato tramite un rettangolo diviso in due parti, da cui escono due segmenti. Il rettangolo è delimitato dal primo e dal terzo quartile, q1/4 e q3/4, e diviso al suo interno dalla mediana, q1/2. I segmenti sono delimitati dal minimo e dal massimo dei valori. In questo modo vengono rappresentati graficamente i quattro intervalli ugualmente popolati delimitati dai quartili.

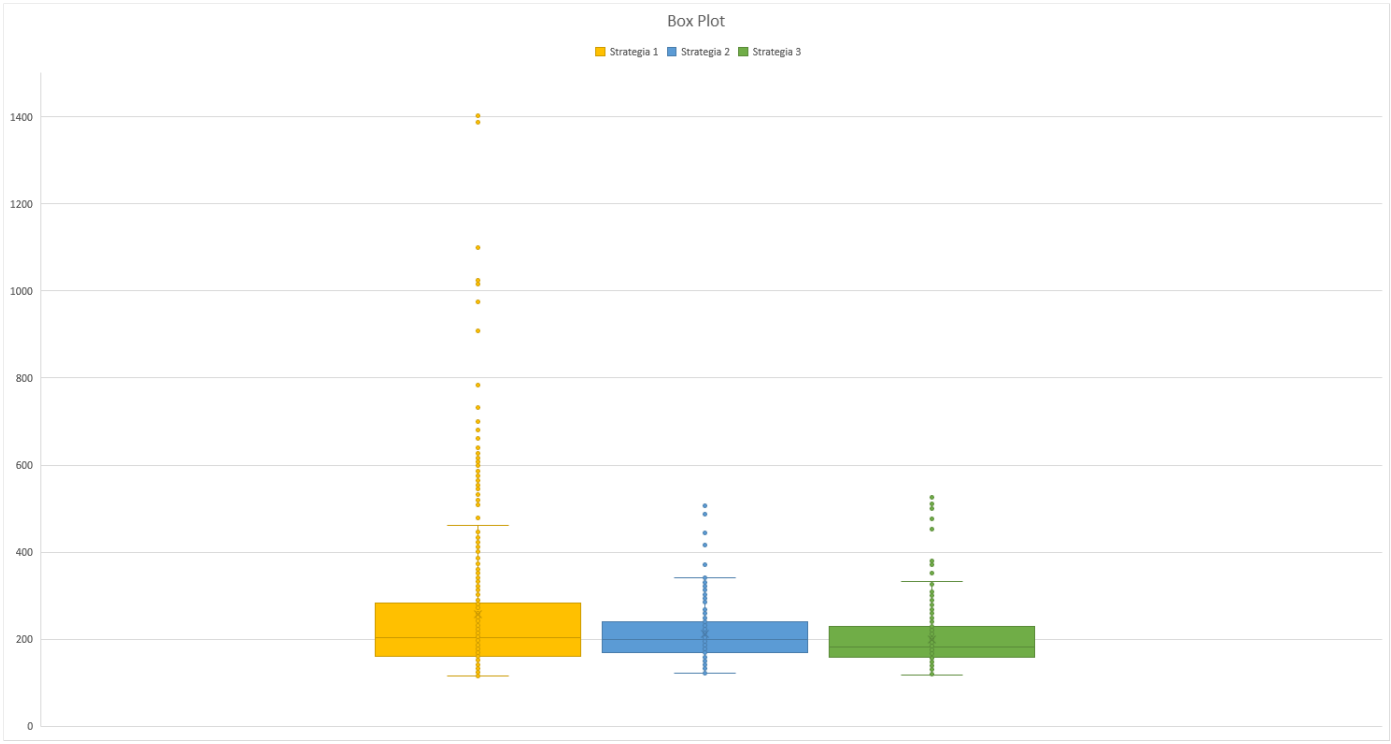


Figura 4.3: Diagramma Scatola e Baffi

Dalla Figura 4.3 si nota come la mediana delle tre Strategie tende a ridursi avvicinandosi alla Strategia 3, ciò significa che la media del tempo di ricerca dei parcheggi tende a ridursi adottando sistemi di Parking Guidance nell'area urbana di San Francisco. I valori estremi, rappresentati dai singoli puntini, nella Strategia 1 arrivano anche a toccare i 1400 secondi, ciò vuol dire che ci sono dei veicoli nella Strategia 1 che impiegano anche 20 minuti per cercare un parcheggio. Nella Strategia 3 invece il valore massimo che si può trovare è circa 450 secondi.

Capitolo 5

Conclusioni

Da questo Capitolo è possibile evincere che effettivamente ci sono delle sostanziali differente tra le tre Strategie, a favore della Strategia 3 è presente un'elevata riduzione dei tempi di ricerca di un parcheggio e di un'altrettanta diminuzione del tragitto totale percorso dai singoli veicoli. Tale studio può essere ampliato considerando anche altri aspetti, come il consumo di carburante o le emissioni di CO2. Tali dati possono essere calcolati o direttamente da SUMO, per un'analisi più accurata, o calcolando in base al consumo/emissioni delle singole vetture per metro percorso.
Lo studio effettuato utilizzando il simulatore è da considerarsi relativamente accurato poichè sono stati considerati diversi aspetti e agenti del traffico, la mappa è stata ottenuta da OpenStreetMap e i dati relativi alla simulazione stessa sono stati offerti direttamente dal comune di San Francisco. Per poter ottenere i dati riguardo i viaggi dei veicoli durante una giornata classica a San Francisco è stato possibile accedere a [23] che offre i dati relativi al flusso di spostamento dei veicoli dai vari quartieri. La sperimentazione è ulteriolmente ampliabile applicandola ad uno scenario ancora più realistico, inserendo all'interno della simulazione altri elementi che possono incidere sui risultati finali. Per esempio è possibile aggiungere i pedoni, gli autobus, i taxi o anche i vigili urbani che incidono sul traffico di determinate aree e possono comportare una modifica dei dati ottenuti e analizzati in questo capitolo.
Il fine di questo studio è stato quello di comprendere, in maniera preliminare, se effettivamente adottare sistemi di Parking Guidance può comportare dei benefici per quanto riguarda il traffico di una determinata area urbana, e se un simulatore come SUMO è capace di poter simulare scenari di questo tipo.
In conclusione SUMO è un ottimo simulatore per scenari microscopici e può rappresentare una soluzione per quanto riguarda il simulare scenari reali su aree geografiche

reali, grazie alla capacità di poter modificare qualsiasi aspetto riguardo il traffico. Nel caso specifico del Rerouting verso aree di parcheggio, il simulatore è ancora migliorabile, inserendo ulteriori algoritmi di Routing e dando la posibilità di poterli attivare senza l'inserimento di parcheggi finti come fatto in questo studio. Un grande vantaggio riscontrato in SUMO, però rappresenta l'architettura TraCI che permette , tramite determinate funzioni presenti nel manuale, di poter personalizzare il comportamento dei veicoli durante la simulazione stessa, rendendo le potenzialità di SUMO pressochè infinite.

5.1 Studi Futuri

Lo studio effettuato in questo elaborato può essere ampliato seguendo diverse strade, è possibile continuare ad effettuare diverse simulazioni su San Francisco per poter raffinare i dati ottenuti e renderli ancora più attendibili, inserendo ulteriori agenti di traffico o effettuando diverse simulazioni dello stesso scenario su *trip.file* differenti, in modo tale da poter effettuare un merge finale dei dati ottenuti.
Un' ulteriore ampliamento dello studio può essere anche quello di applicare le stesse Strategie viste per San Francisco su altre aree urbane di diverse dimensioni e più o meno popolate. Ovviamente i dati ottenuti sarebbero diversi da quelli di San Francisco e comprendere se utilizzare sistemi di Parking Guidance può non risultare molto conveniente su aree urbane con conformazione differente è un aspetto da considerare per poter valutare la bontà di tali tecnologie al 100%.

Elenco delle figure

Bibliografia

[1] V Bharathi et al. "Smart parking guidance system using RASPBERRY-PI". In: *Materials Today: Proceedings* (2020).

[2] "Digital 2022: Global Overview Report". In: *DataReportal-Global Digital Insights* (2022).

[3] *Wikipedia - Parking Guidance Information*. URL: `https://en.wikipedia.org/wiki/Parking_guidance_and_information`.

[4] Kay W Axhausen et al. "Effectiveness of the parking guidance information system in Frankfurt am Main". In: *Traffic engineering & control* 35.5 (1994), pp. 304–309.

[5] Fabian Bock, Jiaqi Liu e Monika Sester. "Learning on-street parking maps from position information of parked vehicles". In: *Geospatial Data in a Changing World*. Springer, 2016, pp. 297–314.

[6] Amir O Kotb, Yao-chun Shen e Yi Huang. "Smart parking guidance, monitoring and reservations: a review". In: *IEEE Intelligent Transportation Systems Magazine* 9.2 (2017), pp. 6–16.

[7] Fabian Bock, Sergio Di Martino e Antonio Origlia. "Smart parking: Using a crowd of taxis to sense on-street parking space availability". In: *IEEE Transactions on Intelligent Transportation Systems* 21.2 (2019), pp. 496–508.

[8] Fabian Bock, Sergio Di Martino e Antonio Origlia. "A 2-step approach to improve data-driven parking availability predictions". In: *Proceedings of the 10th ACM SIGSPATIAL workshop on computational transportation science*. 2017, pp. 13–18.

[9] Mingkai Chen e Tianhai Chang. "A parking guidance and information system based on wireless sensor network". In: *2011 IEEE International Conference on Information and Automation*. IEEE. 2011, pp. 601–605.

[10] *Spotter - l'app del parcheggio*. URL: `https://www.spotterapp.it/`.

[11] Suhas Mathur et al. "Parknet: drive-by sensing of road-side parking statistics". In: *Proceedings of the 8th international conference on Mobile systems, applications, and services*. 2010, pp. 123–136.

[12] Geert Tasseron, Karel Martens e Rob van der Heijden. "The potential impact of vehicle-to-vehicle communication on on-street parking under heterogeneous conditions". In: *IEEE Intelligent Transportation Systems Magazine* 8.2 (2016), pp. 33–42.

[13] Itzhak Benenson, Karel Martens e Slava Birfir. "PARKAGENT: An agent-based model of parking in the city". In: *Computers, Environment and Urban Systems* 32.6 (2008), pp. 431–439.

[14] Adam Millard-Ball, Rachel R Weinberger e Robert C Hampshire. "Is the curb 80% full or 20% empty? Assessing the impacts of San Francisco's parking pricing experiment". In: *Transportation Research Part A: Policy and Practice* 63 (2014), pp. 76–92.

[15] Mark Wardman, V Phani K Chintakayala e Gerard de Jong. "Values of travel time in Europe: Review and meta-analysis". In: *Transportation Research Part A: Policy and Practice* 94 (2016), pp. 93–111.

[16] *Drivers spend an average of 17 hours a year searching for parking spots.* URL: https://eu.usatoday.com/story/money/2017/07/12/parking-pain-causes-financial-and-personal-strain/467637001/.

[17] *Gli italiani perdono 30 minuti giorno in cerca parcheggio.* URL: https://www.agi.it/cronaca/news/2022-04-30/automobilisti-italiani-perdono-30-minuti-giorno-parcheggio-16553725/.

[18] *SUMO Documentation.* URL: https://sumo.dlr.de/docs/index.html.

[19] *TraCI.* URL: https://sumo.dlr.de/docs/TraCI.html.

[20] *Python.* URL: https://www.python.org/.

[21] *OpenStreetMap.* URL: https://www.openstreetmap.org/about.

[22] Huajun Chai e Caroline Rodier. "Automated Vehicles and Central Business District Parking: The Effects of Drop-off-Travel on Traffic Flow and Vehicle Emissions [supporting dataset]". In: (2020).

[23] *ConnectSF, trips information.* URL: https://connectsf-trippatterns.sfcta.org/.