



UNIVERSITÀ DI TRENTO

DIPARTIMENTO DI INGEGNERIA E SCIENZA DELL'INFORMAZIONE

LAUREA TRIENNALE IN INFORMATICA

~ · ~

ANNO ACCADEMICO 2023-2024

Design e analisi di una manovra di sorpasso cooperativo in ambiente simulato

Relatore

Prof. Michele SEGATA

Studente

Ioan Gabriel DUTA

227818

Correlatore

Prof. Paolo CASARI

DATA ESAME FINALE: 26 Novembre 2024

Indice

Sommario	1
1 Introduzione	2
1.1 Definizione e contesto	2
1.2 Obiettivi e vantaggi	2
1.2.1 Sicurezza	2
1.2.2 Efficienza del traffico	3
1.2.3 Riduzione dei consumi	4
1.3 Tassonomia	4
1.4 Case-study: Platooning	5
1.5 Sfide e prospettive future	6
2 Simulazione: software e framework	8
2.1 OMNeT++	8
2.1.1 Simulatore di eventi discreti	8
2.1.2 Moduli	9
2.1.3 File NED	9
2.1.4 File di configurazione e risultati	10
2.2 SUMO	10
2.3 Veins	11
2.4 Plexe	11
3 Manovra di sorpasso	13
3.1 Definizione problema	13
3.2 Possibili casistiche	13
3.3 Algoritmo decisionale	14
3.3.1 Decisione manovra	16
3.3.2 Esempio numerico	17
4 Simulazione e risultati	19
4.1 Setup simulazione	19
4.1.1 SUMO	19
4.1.2 OMNeT++	19
4.1.3 Plexe	20
4.2 Parametri della simulazione	22
4.3 Risultati	22
4.3.1 Emissioni	22
4.3.2 Tempo di sorpasso	23
4.3.3 Analisi dei casi	24
4.3.4 Limiti del contributo	24
5 Conclusioni	26
5.1 Limitazioni	26
5.2 Lavori futuri	26
Bibliografia	27

Elenco delle figure

1.1	Esempio di sistema distribuito [7]	4
1.2	Illustrazione di un platoon in azione [7]	6
1.3	Throughput stradale misurato in un ring di 10km, per platoon di 8 veicoli. La figura mostra anche il throughput teorico massimo (free-flow) e quello usato come base (veicoli che usano solo ACC) [5].	6
2.1	Diagramma dei software e framework in gioco e delle loro interazioni [10].	8
2.2	Flow-chart generico di un simulatore di eventi discreti [9]	9
2.3	Struttura dei modelli in OMNeT++. Le frecce che connettono i modelli rappresentano connessioni e porte [14].	9
2.4	Esempio di file NED preso dalla manovra di sorpasso.	10
2.5	Simulazione nella GUI di SUMO e un grafico associato.	10
2.6	Definizione delle classi Veins TraCICommandInterface e TraCICommandInterface::Vehicle.	11
2.7	Struttura del framework di simulazione Plexe [11].	11
2.8	Esempio di inizializzazione e utilizzo delle API di Plexe.	12
3.1	Disposizione dei platoon nella simulazione.	13
3.2	Scenario con dati e misure utili.	14
4.1	Flowchart della simulazione.	20
4.2	Parte di file NED del modulo traffic.	20
4.3	ManeuverMessage.	21
4.4	(Sinistra) Emissioni medie aggregate per distanza di partenza del Platoon C (Destra) Emissioni medie aggregate per velocità di partenza del Platoon C	23
4.5	Tempi medi di completamento manovra per (Sinistra) distanza di partenza e (Destra) velocità di partenza del Platoon C	23
4.6	Scelta presa dall'algoritmo decisionale per la manovra con rallentamento del Platoon C a disposizione.	24

Sommario

Questa tesi affronta l'implementazione della manovra cooperativa di sorpasso in un ambiente simulato. Nel primo capitolo viene introdotto il concetto di manovre cooperative e i benefici che apportano in termini di sicurezza, efficienza del traffico e riduzione dei consumi. Viene approfondito il concetto di "platooning" come esempio di guida cooperativa, in cui più veicoli si coordinano per viaggiare insieme con distanze ridotte, ottimizzando così i consumi e la fluidità del traffico.

Successivamente, nel secondo capitolo, si descrivono i principali strumenti di simulazione utilizzati evidenziando come questi software interagiscono per modellare e gestire la simulazione della manovra.

Il terzo capitolo analizza la manovra di sorpasso, descrivendo i problemi da affrontare e l'algoritmo decisionale sviluppato per valutare se e quando effettuare il sorpasso. Vengono presentate diverse casistiche (sorpasso immediato, attesa veicolo opposto e con rallentamento del veicolo opposto) e l'approccio matematico adottato per la coordinazione tra i veicoli in diverse situazioni.

Nel quarto capitolo vengono esposti i risultati delle simulazioni. Confrontando le performance in termini di emissioni di CO₂ e tempi di completamento delle manovre, si evidenzia come la cooperazione tra veicoli riduca sia le emissioni che i tempi della manovra, in particolare nei casi in cui uno dei veicoli rallenta per facilitare il sorpasso.

1 Introduzione

Le nostre strade sono sempre più popolate da auto a guida autonoma, una volta adottate su larga scala una naturale evoluzione sarà implementare meccanismi di cooperazione e comunicazione reciproca. Questo nuovo paradigma permetterà ai veicoli non solo di percepire in modo più completo l'ambiente circostante, ma di comunicare tra di loro per coordinare le proprie azioni. Lo scopo è permettere ai veicoli di affrontare in modo più efficace situazioni complesse, come il traffico denso o le manovre pericolose, riducendo i rischi legati all'imprevedibilità dei singoli veicoli e ottimizzando i flussi di traffico. La guida cooperativa andrà a sfruttare tecnologie di comunicazione veicolo-veicolo (V2V) e veicolo-infrastruttura (V2I) per raggiungere questi obiettivi.

Un esempio di guida cooperativa che verrà anche utilizzato in questo capitolo è il platooning: un gruppo di veicoli che viaggia nella stessa direzione può formare un treno stradale, seguendo in modo automatico l'un l'altro. Da questo esempio potremo vedere nella pratica tutti i vantaggi che la cooperazione tra veicoli può portare.

1.1 Definizione e contesto

I veicoli autonomi per prendere le proprie decisioni si basano su osservazioni e ragionamenti sviluppati in maniera individuale, scegliendo azioni che ottimizzano solamente i propri obiettivi. Si sta riconoscendo sempre di più che, per ottenere il massimo dei vantaggi dalla guida autonoma, diverse situazioni richiedono il coordinamento delle relative azioni e delle manovre dei veicoli.

Le manovre cooperative vanno intese come azioni coordinate tra veicoli che si basano sulla comunicazione esplicita per prendere decisioni basate su informazioni più complete.

1.2 Obiettivi e vantaggi

Gli obiettivi cruciali delle manovre cooperative vanno ad affrontare problemi molto attuali nella nostra mobilità moderna.

1.2.1 Sicurezza

La condivisione di informazioni in tempo reale tra veicoli e infrastrutture migliora la consapevolezza situazionale del veicolo, consentendo di individuare potenziali pericoli in anticipo e di adottare misure preventive. Prendendo spunto dalla fonte [9] possiamo evidenziare diverse situazioni.

Sicurezza stradale

Importanti sono i casi in cui bisogna prestare particolare attenzione agli utenti vulnerabili della strada (VRU) come pedoni, ciclisti e motociclisti, che sono particolarmente esposti al rischio di incidenti, la comunicazione tra veicoli può svolgere un ruolo fondamentale nel miglioramento della loro sicurezza. Ad esempio, un sistema di comunicazione potrebbe avvisare un automobilista della presenza di un ciclista in un punto cieco o segnalare l'attraversamento di un pedone.

Una situazione menzionata è il rischio per i pedoni che scendono da un autobus e attraversano la strada. I bambini, in particolare, potrebbero non essere pienamente consapevoli del pericolo e attraversare senza prestare sufficiente attenzione. Un sistema di comunicazione V2I potrebbe avvisare gli automobilisti in avvicinamento, prevenendo potenziali incidenti.

Spostandosi a casi di scarsa visibilità o di visibilità limitata (si pensi ad edifici o altri ostacoli che possono ostruire la vista) unire la conoscenza ambientale che un veicolo percepisce con i propri sistemi a quella di altri veicoli permetterà di valutare più accuratamente le azioni da eseguire proprio grazie alla presenza di più dati utili. Ad esempio nella manovra che vedremo in seguito, il sorpasso, la presenza di un veicolo in arrivo potrebbe non essere percepita da un veicolo che sta effettuando la

manovra, ma potrebbe essere comunicata da un veicolo che si trova in posizione migliore e che è in grado di identificarlo.

Anche eventuali situazioni di emergenza possono essere comunicate in maniera tempestiva. Ad esempio in caso di una frenata improvvisa o un incidente la comunicazione V2V permetterà di avvertire gli altri veicoli della situazione in maniera più veloce ed efficace per evitare collisioni. Oppure la presenza di detriti sulla carreggiata se comunicata in tempo reale permetterebbe agli altri veicoli di adottare misure preventive.

Sicurezza informatica

Queste nuove tecnologie ci espongono anche a nuove sfide a livello di sicurezza informatica. È fondamentale garantire che questi sistemi siano protetti da attacchi hacker che compromettano il loro funzionamento e mettano a rischio la sicurezza degli occupanti del veicolo e di altri utenti della strada. Alcune sfide specifiche sono:

- **Autenticazione e autorizzazione:** garantire che solo i dispositivi autorizzati possano accedere alla rete veicolare e inviare comandi. Sistemi di autenticazione robusti sono necessari per prevenire attacchi di spoofing e man-in-the-middle.
- **Integrità dei messaggi:** assicurare che i messaggi scambiati tra i veicoli non siano stati alterati o manomessi per evitare che un attore maligno li utilizzi per creare disagi o incidenti. Algoritmi di crittografia e firme digitali possono essere utilizzati per proteggere i messaggi da alterazioni non autorizzate.
- **Riservatezza dei dati:** dati scambiati tra veicoli, come la posizione, la velocità e la direzione, possono essere sensibili e devono essere protetti da accessi non autorizzati per evitare il tracciamento e l'identificazione degli occupanti da persone non autorizzate. La crittografia dei dati e l'anonymizzazione delle identità dei veicoli possono contribuire a preservare la privacy degli utenti.

1.2.2 Efficienza del traffico

Un altro importante risultato della comunicazione tra veicoli è una migliore gestione del traffico. In tutto il mondo le infrastrutture stradali affrontano diversi problemi di traffico tra cui congestione, scarsa sicurezza e incidenti stradali. Per fronteggiare questi problemi l'idea generale è di costruire nuove infrastrutture, il che risulta sempre meno pratico per le grandi città che stanno finendo lo spazio a disposizione. Un'altra soluzione sempre più popolare è quella di ampliare la capacità delle strade, sfruttando infrastrutture di sistemi di trasporto intelligente (ITS) e i loro sottocomponenti. È evidenziato dalla ricerca che gli ITS possono massimizzare la capacità delle strade fino al 20%. Il più importante componente di questi ITS è proprio la guida collaborativa [7].

Vedremo più in dettaglio come le manovre cooperative, ad esempio il platooning, portano ad una riduzione delle congestioni e dell'utilizzo delle infrastrutture, liberando le amministrazioni dalla necessità di costruirne di nuove [5].

Un altro aspetto importante è il miglioramento dell'esperienza del conducente e dei passeggeri, anche in casi di guida non completamente autonoma: [9]

- **Riduzione dello stress:** alcune funzioni tediose come il mantenimento della distanza di sicurezza o il cambio di corsia possono essere automatizzate dai sistemi di guida cooperativa.
- **Maggiore confort:** la guida in platoon può risultare più confortevole rispetto alla guida tradizionale, poiché i veicoli mantengono una velocità e una distanza costanti, riducendo le oscillazioni e le frenate improvvise.
- **Nuove funzionalità:** la comunicazione tra veicoli può consentire l'implementazione di nuove funzionalità, come l'infotainment condiviso, l'accesso a informazioni sul traffico in tempo reale e la comunicazione con i servizi di emergenza.

1.2.3 Riduzione dei consumi

Dal capitolo precedente possiamo intuire come il miglioramento del traffico porti anche una diminuzione generale dei consumi.

Sempre prendendo esempio dalla fonte [9] forniamo alcune cause specifiche di questo miglioramento:

- **Anticipazione delle variazioni di velocità:** evitare frenate brusche e accelerazioni improvvise causa meno emissioni. La cooperazione consente ai veicoli di adattare la propria velocità in modo che la guida risulti più fluida.
- **Resistenza aerodinamica:** i veicoli che viaggiano in platoon beneficiano di una riduzione della resistenza aerodinamica, che a sua volta si traduce in un minor consumo di carburante. Questo è particolarmente rilevante per i camion, ma può portare benefici anche alle autovetture.
- **Ottimizzazione dei percorsi:** sistemi di navigazione che sfruttano la comunicazione V2I possono suggerire percorsi che minimizzano il consumo di carburante, sfruttando dati ottenuti in tempo reale, come le condizioni del traffico o incidenti appena avvenuti.

Nel capitolo dedicato ai risultati della manovra di sorpasso vedremo nel pratico come pattern di guida ottenibili solo attraverso la cooperazione portino ad una minore emissione di CO₂.

1.3 Tassonomia

Questa sezione offre una visione generale della tassonomia della guida autonoma cooperativa. Come possiamo vedere in Figura 1.1 la tassonomia proposta è divisa in 2 categorie [7]:

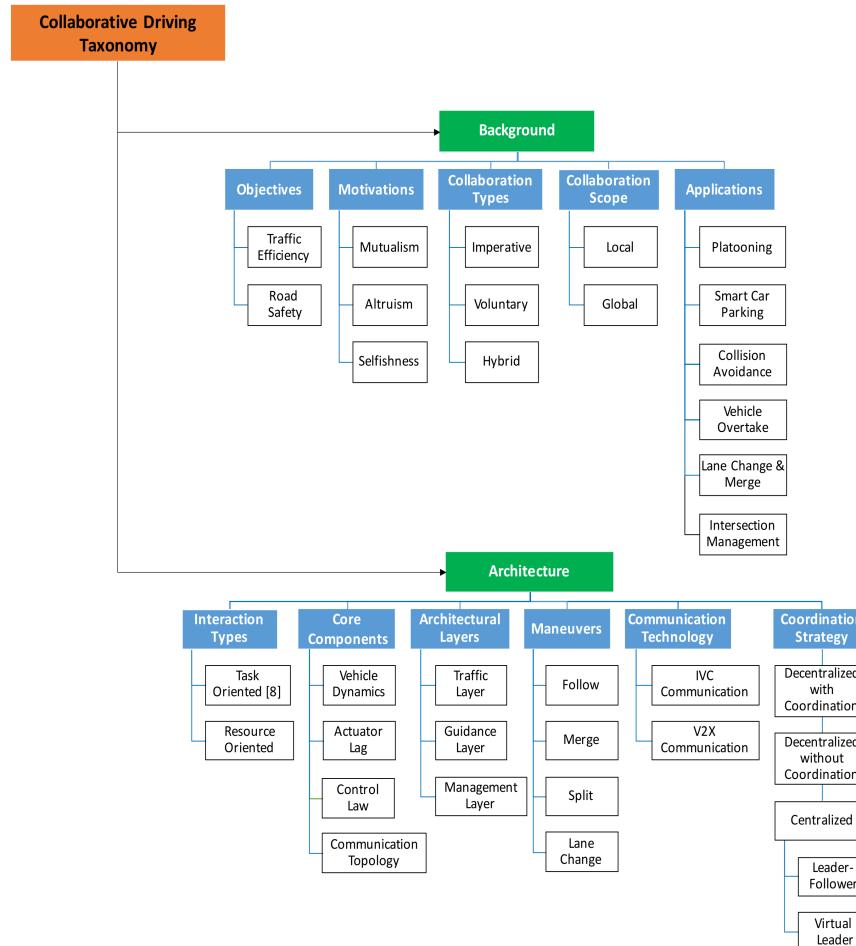


Figura 1.1: Esempio di sistema distribuito [7]

- Background: comprende obiettivi, motivazioni, tipi di collaborazione, livelli di collaborazione e applicazioni.
- Architettura: include tipi di interazione, componenti principali, strati dell'architettura, manovre, tecnologie per la comunicazione e strategie di coordinamento.

Andiamo a vedere alcune di queste categorie più nel dettaglio [7]:

1. Perché un veicolo dovrebbe collaborare? È cruciale capire perché un veicolo dovrebbe essere disposto a collaborare:

- Mutualismo: gli agenti cooperano e interagiscono tra loro per minimizzare il costo (carburante, tempo di viaggio, etc.) di tutti i veicoli collettivamente.
- Altruismo: un agente interagisce con gli altri aumentando il proprio costo per ridurre il costo degli altri. Ne vedemo una esempio pratico nella manovra di sorpasso studiata in seguito.
- Opportunismo: il tipo di interazione in cui l'agente cerca di diminuire il proprio costo lasciando gli altri in perdita.

2. Strategie di controllo della cooperazione: per ottenere un'organizzazione degli agenti ci sono 2 diverse strategie di coordinamento:

- Coordinazione centralizzata: un veicolo leader, a conoscenza di tutti i dati, responsabile della pianificazione, coordinazione e sincronizzazione delle manovre di tutti i veicoli. La letteratura divide questo tipo di strategia in 2 varianti: leader-follower e leader virtuale. La manovra di sorpasso utilizzerà questa strategia. Un leader valuterà la fattibilità della manovra e coordinerà i propri follower e, in alcuni casi, gli altri platoon per ottimizzare al meglio il sorpasso.
- Decentralizzata con coordinamento: permette ai veicoli di comunicare direttamente con i veicoli vicini accedendo a informazioni locali.

3. Tipologie di collaborazione: la cooperazione può essere classificata in base al livello di interazione tra i veicoli:

- Collaborazione imperativa: il tipo di collaborazione in cui i veicoli sono obbligati a cooperare.
- Collaborazione volontaria: per gli agenti è opzionale cooperare, dipende dallo use-case.
- Collaborazione ibrida: permette ad un agente di collaborare imperativamente per un certo periodo e volontariamente finire la collaborazione.

1.4 Case-study: Platooning

Il platooning è un use-case importante per la guida cooperativa. Ha ricevuto molta attenzione per i potenziali benefici che offre nell'immediato, come maggiore sicurezza stradale, riduzione dei consumi e della CO₂ [7] e riduzione dello stress del conducente: questo è particolarmente importante per professionisti e pendolari. [5]. Il platooning è una tecnologia che consente ad un convoglio di più veicoli autonomi di viaggiare in formazione, uno dietro l'altro, mantenendo una distanza ridotta e sincronizzando movimenti e velocità, comunicando tra loro.

Distinguiamo anche due diversi livelli del platooning [4]:

- **Platooning control:** il controllo del singolo veicolo al livello più base possibile. Include il mantenimento della distanza, l'invio di segnali di frenata o qualiasi segnalazione inviabile agli altri membri del platoon.
- **Platooning coordination:** include il management di (i) la composizione del platoon, (ii) le interazioni inter-platoon e (iii) l'interazione con altri veicoli e platoon.

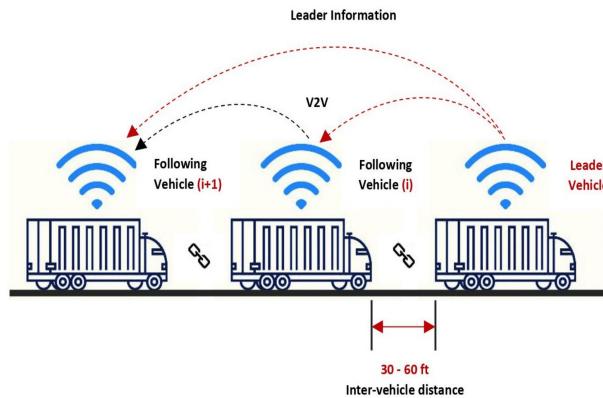


Figura 1.2: Illustrazione di un platoon in azione [7].

Nella sua forma più semplice, per il platooning, potremmo usare lo standard ACC (Adaptive Cruise Control), già disponibile sui nuovi veicoli, il quale misura la distanza e la velocità relativa del veicolo che precede e calcola l'accelerazione necessaria per mantenere la "constant time headway", il tempo che intercorre tra il passaggio di due veicoli consecutivi per un punto nello spazio. In realtà l'ACC richiede una distanza di headway molto grande per garantire stabilità del platoon e sicurezza.

Per questo motivo sono stati sviluppati algoritmi di CACC (Cooperative ACC), i quali estendono l'ACC prendendo in considerazione altri dati ottenuti attraverso la comunicazione.

I veicoli, grazie alla comunicazione, non avranno solo dati riguardanti l'ambiente al di là del loro campo visivo, ma saranno in grado di condividere le proprie intenzioni e le future azioni che vogliono compiere, rendendo semplice e affidabile la previsione.

L'idea principale è ottenere informazioni dal veicolo che precede e dal leader del platoon, può essere dimostrato che considerando questi dati il platoon è sicuro e stabile anche seguendo una policy di spacing costante.

Ricollegandoci al capitolo 1.2.2 possiamo vedere in Figura 1.3 i risultati in una simulazione che studia il throughput su una strada di 10km. Si può vedere come diversi algoritmi di CACC adottati dal 25% o 75% dei veicoli portino ad un migliore throughput rispetto ad una simulazione in cui tutti i veicoli utilizzano l'ACC [5].

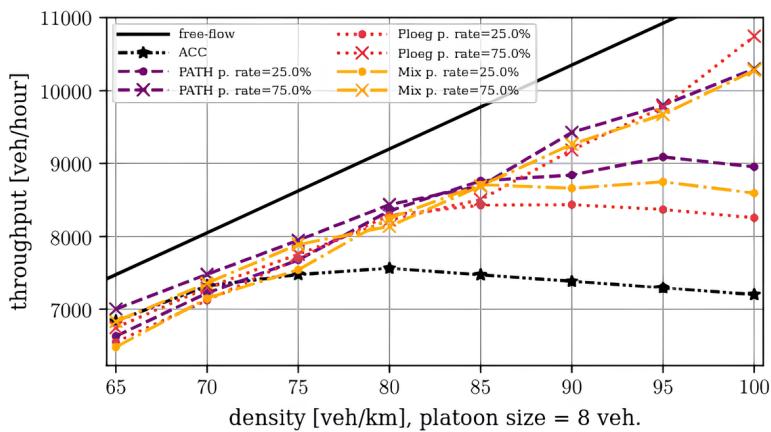


Figura 1.3: Throughput stradale misurato in un ring di 10km, per platoon di 8 veicoli. La figura mostra anche il throughput teorico massimo (free-flow) e quello usato come base (veicoli che usano solo ACC) [5].

1.5 Sfide e prospettive future

Abbiamo visto molti vantaggi che la guida cooperativa può offrire, ma ci sono anche diverse sfide da affrontare. Alcune di queste sono:

- **Comunicazione:** bisognerà sviluppare tecnologie per la comunicazione V2V assicurandosi che siano affidabili, robuste e sicure.
- **Ottimizzazione multi-obiettivo:** i framework di guida collaborativa devono considerare sia gli obiettivi globali (ad esempio, evitare collisioni e migliorare l'efficienza del traffico) sia gli obiettivi locali dei singoli veicoli, come preferenze, obiettivi e vincoli individuali. Se la collaborazione non si traduce in un beneficio per tutti i veicoli coinvolti, la loro propensione a collaborare potrebbe diminuire.
- **Cooperazione eterogenea:** i veicoli essendo di diverse marche e tipologie monteranno sicuramente sistemi diversi. È necessario definire standard di comunicazione e protocolli per garantire l'interoperabilità tra veicoli eterogenei.
- **Comportamento egoistico:** alcuni veicoli potrebbero cercare di sfruttare i vantaggi della cooperazione senza contribuire attivamente, saranno necessari sistemi e algoritmi in grado di rilevare questi comportamenti egoistici e potenzialmente escludere i veicoli non cooperativi.
- **Scala della collaborazione:** è impossibile far collaborare tutti i veicoli in una particolare area, come una città, in quanto i dati da analizzare sarebbero troppi. Bisognerà porsi delle domande su quale sia il numero minimo di veicoli necessario per ottenere i benefici della cooperazione e fino a che estensione possono collaborare.

Infine nella valutazione della guida cooperativa bisogna considerare anche l'aspetto umano; sarà necessario fornire una maggiore consapevolezza al pubblico sui vantaggi e sulle modalità di funzionamento di questa tecnologia, per garantire la sua accettazione e adozione su larga scala. Come vedremo nella manovra di sorpasso i margini di sicurezza a cui è abituato l'uomo possono essere diminuiti a distanze molto ridotte. Quanto ci vorrà affinché gli utenti si abituino e accettino queste nuove situazioni?

2 Simulazione: software e framework

In questo capitolo presenteremo e descriveremo brevemente i software e i framework utilizzati per creare la simulazione della manovra di sorpasso. I protagonisti principali sono **Plexe** che offre le API per gestire i veicoli e **Sumo** che si occupa della simulazione del traffico stradale e permette di visualizzare in una GUI la manovra e altri dati utili. Come motore di tutta la simulazione abbiamo **OMNeT++**, un ambiente di simulazione di eventi discreti. Infine **Veins**, costruito sopra OMNeT++, permette di simulare la comunicazione veicolo-veicolo e veicolo-infrastruttura.

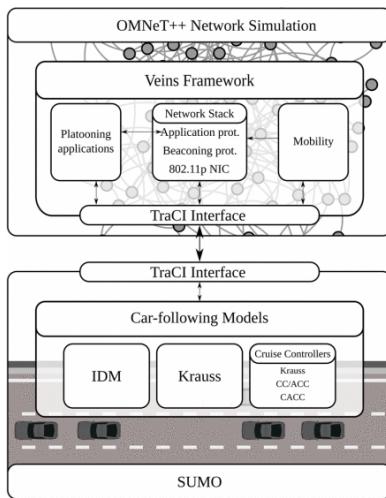


Figura 2.1: Diagramma dei software e framework in gioco e delle loro interazioni [10].

2.1 OMNeT++

OMNet++ è un simulatore di eventi discreti, basato su C++, creato per modellare sistemi distribuiti o paralleli, tra cui reti di comunicazione. OMNeT++ non è un simulatore single-purpose, ma offre un approccio framework per la creazione di componenti destinati a diversi scopi mettendo a disposizione meccanismi di base e tool per crearli [14].

2.1.1 Simulatore di eventi discreti

OMNeT++ è un simulatore che modella il comportamento di un sistema come una sequenza di eventi che avvengono in istanti discreti. Nelle simulazioni discrete ad eventi l'evoluzione della simulazione è guidata dagli stessi. Un evento può essere qualsiasi cosa che cambia lo stato del sistema, come l'invio di un pacchetto da un nodo di rete, la ricezione di un pacchetto, un timeout TCP, etc. Ogni evento è associato a un istante di tempo e a un modulo specifico che deve gestirlo. Il simulatore mantiene una coda di eventi ordinata per tempo e processa gli eventi uno alla volta, in ordine cronologico.

Ad esempio, se si simula il movimento di un'auto e si imposta un tempo di campionamento di 1 secondo, la simulazione fornirà la posizione dell'auto a 1 secondo, 2 secondi, 3 secondi, e così via, ma non a 1,5 secondi o 2,7 secondi [9].

Un concetto chiave nella simulazione discreta ad eventi è che gli eventi sono atomici. Ciò significa che durante l'elaborazione di un evento il tempo di simulazione si ferma.

In Figura 2.2 vediamo un flow-chart generico del funzionamento di un simulatore di eventi discreti.

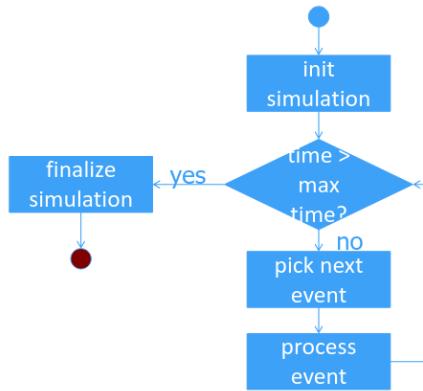


Figura 2.2: Flow-chart generico di un simulatore di eventi discritti [9]

2.1.2 Moduli

Una simulazione OMNeT++ consiste di moduli che comunicano tra loro scambiandosi messaggi. Questi sono scritti in C++ usando le classi della libreria di simulazione. L'unità più piccola sono i moduli di base, questi possono essere raggruppati in moduli composti e così via, senza un limite al numero di livelli. Infine abbiamo anche moduli composti definiti come *networks*; questi sono moduli di simulazione auto-contenuti.

Generalmente i moduli di base inviano messaggi attraverso porte o direttamente al modulo di destinazione; questi messaggi, oltre ai soliti attributi, come il timestamp, possono contenere dati arbitrari.

In Figura 2.3 vediamo un semplice esempio di gerarchia di moduli [14].

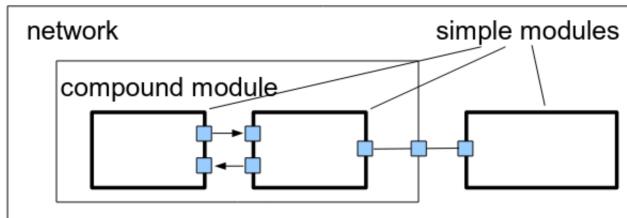


Figura 2.3: Struttura dei modelli in OMNeT++. Le frecce che connettono i modelli rappresentano connessioni e porte [14].

I moduli possono avere *parametri*, questi sono generalmente utilizzati per passare dati di configurazione agli stessi, e per definirne la topologia. I parametri possono essere inizializzati con stringhe, numeri o valori booleani. Vediamo come si possono definire [14].

2.1.3 File NED

Il linguaggio messo a disposizione da OMNeT++ per descrivere la topologia e la struttura di un modulo e delle interconnessioni tra moduli è il linguaggio NED (Network Description). Questo linguaggio permette all'utente di dichiarare moduli, connetterli e assemblarli in moduli composti.

Il linguaggio NED ha diverse features che gli permettono di scalare facilmente: è *gerarchico*; è *component-based*: i moduli sono riutilizzabili, riduce la quantità di codice e permette la creazione di librerie di componenti; mette a disposizione *interfacce*: utilizzate come placeholder al posto di moduli o canali determinati poi a network set-up time da un parametro; offre *inheritance*: moduli e canali possono ereditare da altre classi; e altre features come *packages*, *inner types* e *metadati* [8].

```

package org.car2x.plexe.apps;

import org.car2x.plexe.apps.BaseApp;

simple OvertakeApp like BaseApp {
    parameters:
        // implementation of the overtake maneuver
        string overtakeManeuver;
    int headerLength @unit("bit") = default(0 bit);
    @display("i=block/app2");
    @class(plexe::OvertakeApp);

    gates:
        input lowerLayerIn;
        output lowerLayerOut;
        input lowerControlIn;
        output lowerControlOut;
}

```

Figura 2.4: Esempio di file NED preso dalla manovra di sorpasso.

2.1.4 File di configurazione e risultati

All'avvio, dopo aver letto i file NED, OMNeT++ esegue il file di configurazione, solitamente chiamato **omnetpp.ini**. Questo file contiene le impostazioni che controllano come la simulazione deve essere eseguita, valori per i parametri dei moduli, opzioni di simulazione, etc. .

I risultati della simulazione invece vengono scritti nei file di output: *vector*, *scalar* e, se definiti, nei file di output dell'utente [8].

2.2 SUMO

SUMO (Simulation of Urban MObility) [6] è un simulatore di traffico stradale *continuo*: per cui lo spazio viene visto con valori continui e non discreti come in altri simulatori; *microscopico*: ogni veicolo all'interno della simulazione viene modellato individualmente e ha definiti posizione e velocità, ad ogni time-step questi valori vengono aggiornati indipendentemente tra i veicoli; *multi-modale*: permette di simulare diversi tipi di utenti della strada, come auto, bus, biciclette, pedoni, etc. [3].

SUMO oltre l'ambiente di simulazione ci mette a disposizione anche una GUI per osservare in tempo reale la simulazione e dati associati ad essa.

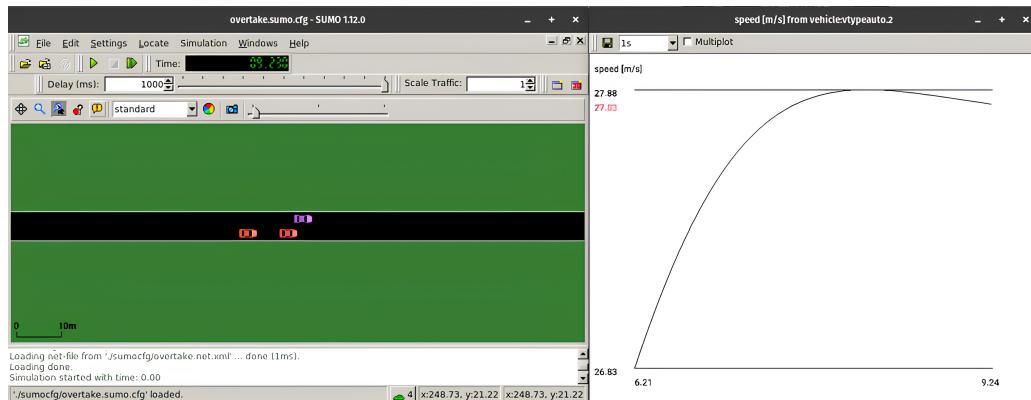


Figura 2.5: Simulazione nella GUI di SUMO e un grafico associato.

Nella Figura 2.5 possiamo notare i comandi per gestire la simulazione come play, pause, stop, etc., comandi per ottenere più informazioni sullo scenario e un grafico che mostra, in questo caso, la velocità di un veicolo in tempo reale.

Infine SUMO ci offre la possibilità di creare scenari personalizzati tramite file di configurazione in formato XML. Nel file *.nex.xml possiamo definire la topologia della strada, ad esempio corsie, giunzioni, etc.; un' altro file che ci interessa è *.rou.xml che definisce i modelli dei veicoli che saranno presenti nella simulazione e i loro parametri.

2.3 Veins

Veins (Vehicular Network Simulation) [13] è un framework creato per supportare simulazioni di veicoli *comunicanti*. Il suo compito principale è stabilire una connessione tra OMNeT++ e un simulatore di traffico stradale, nel nostro caso SUMO, attraverso la **Traffic Control Interface** (TraCI) messa a disposizione da quest'ultimo.

Grazie a TraCI Veins ci permette di controllare la simulazione, ottenendo informazioni su ogni oggetto presente e cambiare i loro attributi.

Quando si crea un nuovo veicolo nella simulazione SUMO, nel nostro caso tramite un inserimento manuale via Veins, viene creato un modulo dedicato nella simulazione OMNeT++, sempre grazie a Veins. Durante la simulazione il veicolo si muove in SUMO e Veins mantiene aggiornato il rispettivo modulo OMNeT++ con le informazioni del veicolo, come posizione, velocità e direzione, insieme ad altri dati. Similmente quando il veicolo raggiunge la sua destinazione in SUMO, Veins rimuove il modulo OMNeT++ associato dalla simulazione [12].

Questa unione tra SUMO e OMNeT++ è bidirezionale, la simulazione OMNeT++ infatti può influenzare la simulazione SUMO inviandole comandi, ad esempio può: far cambiare rotta ai veicoli oppure far eseguire una frenata d'emergenza in risposta ad un avvertimento ricevuto [12].

La comunicazione OMNeT++ → SUMO viene eseguita chiamando metodi dall'interfaccia **TraCICommandInterface** e classi relative messe a disposizione da Veins. In Figura 2.6 vediamo un esempio di come si possano inizializzare queste classi preso direttamente dal codice della manovra di sorpasso.

```
mobility = veins::TraCIMobilityAccess().get(getParentModule());
traci = mobility->getCommandInterface();
traciVehicle = mobility->getVehicleCommandInterface();
```

Figura 2.6: Definizione delle classi Veins TraCICommandInterface e TraCICommandInterface::Vehicle.

2.4 Plexe

Plexe [11] è un framework per la guida cooperativa che estende SUMO e Veins, permettendo la simulazione realistica di sistemi di platooning. Plexe offre veicoli con dinamica realistica, diversi modelli di cruise control e manovre cooperative.

Costruendo sui moduli di Veins, Plexe sfrutta ed estende l'interfaccia a TraCI, permettendo lo scambio di informazioni di stato nelle simulazioni SUMO. Questo include il trasferimento di dati ai sistemi di controllo dei veicoli (come informazioni sui veicoli vicini utilizzati per le azioni di controllo, ne vedremo un esempio nella manovra di sorpasso) o il recupero di dati di un veicolo (informazioni da mandare ai veicoli vicini) [11].

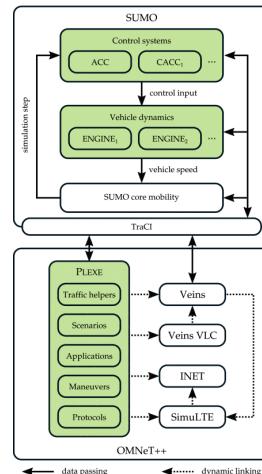


Figura 2.7: Struttura del framework di simulazione Plexe [11].

In Figura 2.7 vediamo che Plexe mette a disposizione tools per implementare manovre, per controllare traffico e platoons, per definire scenari, protocolli e applicazioni [11].

Inoltre Plexe offre diverse API per gestire queste funzionalità, ne vediamo un esempio in Figura 2.8. Ne saranno osservate altre nel capitolo dedicato alla manovra di sorpasso.

```
auto plexe = FindModule<PlexeManager*>::findGlobalModule();
plexeTraci = plexe->getCommandInterface();
plexeTraciVehicle.reset(new traci::CommandInterface::Vehicle(plexeTraci, mobility->getExternalId()));

plexeTraciVehicle->getVehicleData(&data);
plexeTraciVehicle->setFixedAcceleration(1, newAcceleration);
```

Figura 2.8: Esempio di inizializzazione e utilizzo delle API di Plexe.

3 Manovra di sorpasso

Il principale contributo di questo elaborato è l'implementazione della manovra di sorpasso nel contesto della guida autonoma cooperativa. Prima di vedere l'implementazione della manovra, analizzeremo il problema del sorpasso, vedremo le possibili situazioni in cui possiamo effettuare la manovra e l'algoritmo decisionale impiegato.

3.1 Definizione problema

L'obiettivo della manovra è far effettuare in sicurezza ad un platoon di veicoli un sorpasso lungo una strada a due corsie a doppio senso di marcia con un platoon soppraggiungente nel senso opposto. Definiamo come *Platoon A* i veicoli che dovranno essere sorpassati; come *Platoon B* il platoon che effettuerà il sorpasso e come *Platoon C* il platoon soppraggiungente.

La sfida principale sarà far capire al *Platoon B* quando e come avviare e concludere la manovra in modo da evitare incidenti. Inoltre, grazie alla comunicazione, sarà possibile coordinare i platoon e valutare se aumentare o diminuire velocità di uno o più veicoli per ottimizzare il tempo totale della manovra e il consumo di carburante. La cooperazione nel sorpasso, così come in altri tipi di manovre, ci permetterà anche di ridurre le distanze di sicurezza tra i veicoli ottenendo così più spazio utile.

Vediamo in Figura 3.1 la struttura della simulazione in SUMO, in *viola* il *Platoon B*, in *giallo* il *Platoon C* e infine in *rosso* il *Platoon A*.

In seguito quando si parlerà di posizioni dei veicoli si farà sempre riferimento alla strada, vista anche come asse delle ascisse, con l'origine nel punto di partenza (a sinistra) della stessa.

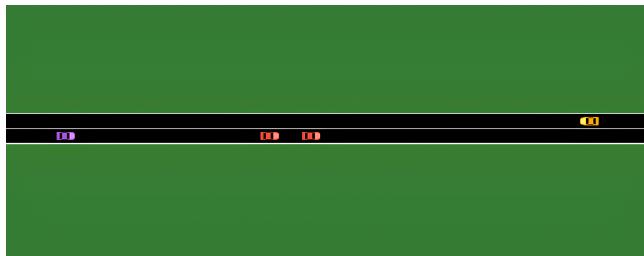


Figura 3.1: Disposizione dei platoon nella simulazione.

In quanto le macchine sono autonome, il *Platoon B* dovrà conoscere posizione e velocità degli altri platoon per poter decidere se effettuare la manovra e come effettuarla, cioè in che momento cambiare corsia e in quale momento tornare nelle propria. Bisognerà valutare i dati e le informazioni che i platoon si scambiano per poter prendere le decisioni necessarie, questo sarà trattato nel prossimo capitolo.

In questa implementazione abbiamo deciso di modificare la velocità solo del *Platoon C* in caso questo porti miglioramenti per la manovra, però è possibile estendere la modifica della velocità anche al *Platoon A* e al *Platoon B*. Tutti i veicoli, anche se singoli, verranno considerati come platoon in quanto la struttura del codice è predisposta ad essere estesa a platoon di più veicoli.

La decisione di effettuare la manovra di sorpasso o meno è presa completamente dal *Platoon B* in quanto diretto interessato, quindi tutte le valutazioni e i calcoli saranno effettuati dal leader di questo; anche lo scambio dei messaggi necessari per la cooperazione sarà gestito da questo.

3.2 Possibili casistiche

Prima di esaminare l'algoritmo decisionale andiamo a vedere le distinte situazioni in cui ci possiamo trovare quando si valuta se effettuare il sorpasso. Dal punto di vista del *Platoon B*, dopo aver calcolato tempo e spazio necessario per il sorpasso, possiamo avere tre situazioni principali:

1. **Sorpasso immediato possibile:** il *Platoon C* è abbastanza lontano da permettere al *Platoon B* di effettuare il sorpasso immediatamente. Questo vuol dire che dopo il tempo necessario per il sorpasso, la posizione del *Platoon C* X_{C_f} meno una distanza di sicurezza d_{head} sarà maggiore della posizione finale del *Platoon B* X_{B_f} , garantendo che i platoon non si possano scontrare.
2. **Sorpasso non possibile:** la posizione finale X_{C_f} del *Platoon C* meno la distanza di sicurezza d_{head} è minore rispetto alla posizione finale X_{B_f} del *Platoon B*. Questo vuol dire che il *Platoon C* andrebbe ad incrociare il *Platoon B* durante la manovra, causando un incidente. Il *Platoon B* dovrà attendere il passaggio del *Platoon C* per effettuare il sorpasso.
3. **Sorpasso possibile con rallentamento del *Platoon C*:** nel caso in cui il sorpasso non sia possibile ci troviamo nella situazione 2. Grazie alla cooperazione possiamo far rallentare il *Platoon C* di un massimo del 10% della sua velocità, e valutare di nuovo in quale delle situazioni ci ritroviamo. Il valore del 10% è stato scelto in modo arbitrario per non far subire ai passeggeri del *Platoon C* eccessivi livelli di decellerazione; può essere modificato.

3.3 Algoritmo decisionale

Quando il leader del *Platoon B* dovrà decidere se effettuare la manovra di sorpasso avrà necessità di sapere posizioni e velocità degli altri platoon. Una volta ottenuti questi dati, che vedremo più nel dettaglio nel prossimo capitolo, l'algoritmo del leader effettuerà dei calcoli per decidere se effettuare la manovra o meno.

Distanze di sicurezza

Prima di tutto definiamo delle variabili che aggiungano delle distanze di sicurezza ai calcoli.

- Utilizzeremo un errore ϵ di qualche metro ogni volta che un veicolo effettua un'azione come accelerazione o decellerazione per considerare i tempi di reazione.
- Definiamo anche una distanza di sicurezza d_{head} che sarà la distanza minima tra due veicoli per evitare collisioni in caso di errori, scelta arbitrariamente e diminuibile. Nelle nostre simulazioni questa è settata a 10 metri.

Variabili principali

Nella Figura 3.2 vediamo lo scenario considerato e alcuni dati e misure che verranno utilizzate nell'algoritmo decisionale.

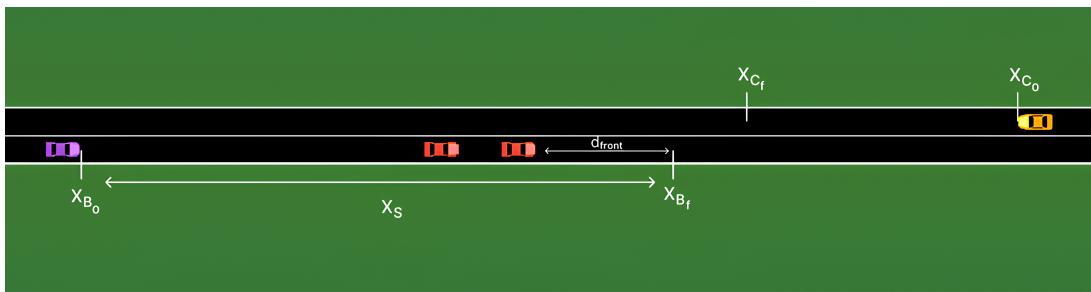


Figura 3.2: Scenario con dati e misure utili.

Andiamo a definire queste variabili:

- d_{front} : distanza di sicurezza tra il *Platoon B* e il *Platoon A* dopo la reimmissione in corsia del platoon B; questa distanza può essere diminuita. Nel nostro caso viene calcolata come una sovrafflata dello spazio di frenata per permettere al *Platoon A* di fermarsi in sicurezza in caso di necessità:

$$d_{front} \equiv \frac{V_A * 3}{10} \text{ m}$$

Dove V_A è la velocità del *Platoon A*.

- X_A : posizione iniziale del *Platoon A*.
- X_{B0} : posizione iniziale del *Platoon B*.
- X_{Bf} : posizione dopo il sorpasso del *Platoon B*.
- X_{C0} : posizione iniziale del *Platoon C*.
- X_{Cf} : posizione a fine sorpasso del *Platoon C*.
- X_S : spazio necessario per effettuare il sorpasso, calcolato come:

$$X_S = X_A - X_{B0} + d_{front}$$

Per effettuare i calcoli saranno necessarie anche le velocità dei platoon, chiamate V_B (velocità iniziale e desiderata dal *Platoon B*), V_{B0} (velocità attuale del *Platoon B*, da prendere in considerazione in quanto a causa del CACC il *Platoon B* rallenta quando si avvicina al *Platoon A*), V_A e V_C rispettivamente. I dati di posizione e velocità del *Platoon A* e del *Platoon C* saranno ottenuti dal leader del *Platoon B* tramite scambio di messaggi.

Tempo per il sorpasso

La prima cosa da valutare è quanto tempo serve al *Platoon B* per sorpassare il *Platoon A*. Dividiamo il calcolo di questo tempo in due parti: una prima fase di accelerazione in cui il *Platoon B* cambia corsia e si riporta alla velocità desiderata V_B , se il sorpasso non è terminato in questa fase allora si considererà la fase finale in cui si porta l'accelerazione del *Platoon B* a 0, quindi la velocità rimane costante, fino alla conclusione del sorpasso.

Per determinare il tempo necessario per la prima fase dobbiamo considerare che la velocità del *Platoon B* potrebbe essere minore di quella desiderata a causa del CACC che fa rallentare il platoon più si avvicina ai veicoli del *Platoon A*. L'obiettivo del CACC è far raggiungere la distanza di headway al *Platoon B* con la stessa velocità del *Platoon A*.

Calcoliamo la differenza di velocità tra il *Platoon B* e il *Platoon A*. Bisogna considerare nel calcolo anche la velocità attuale del *Platoon B*:

$$\Delta V_{BA} = \frac{(V_B - V_A) + (V_{B0} - V_A)}{2} \frac{m}{s}$$

Ottieniamo così la velocità media del *Platoon B* relativa al *Platoon A*.

Ora calcoliamo il tempo necessario al *Platoon B* per raggiungere la velocità desiderata V_B :

$$t_{accel} = \frac{V_B - V_{B0}}{a_B} s$$

Dove a_B è 1.5, l'accelerazione del platoon data dal modello del veicolo definito in SUMO. Questo è un parametro è può essere modificato nel file di configurazione SUMO `*.rou.xml`.

La distanza totale coperta durante la prima fase sarà quindi:

$$d_{accel} = \Delta V_{BA} \cdot t_{accel} m$$

Se d_{accel} è minore di X_S allora il *Platoon B* non ha ancora coperto la distanza necessaria per effettuare il sorpasso.

Raggiunta ora la velocità desiderata V_B non sarà più necessario accellerare ($a_B = 0$).

Calcoliamo quindi lo spazio rimanente da coprire:

$$d_{rest} = X_S - d_{accel} m$$

E il tempo necessario per coprire questo spazio:

$$t_{const} = \frac{d_{rest}}{V_B - V_A} \text{ s}$$

Infine possiamo unire il tutto per ottenere il **tempo totale** necessario per effettuare il sorpasso, dato fondamentale che utilizzeremo anche per determinare la posizione finale del *Platoon C*, sarà anche utilizzato nel prossimo capitolo nella simulazione per programmare il rientro in corsia del *Platoon B*:

$$t_{sorpasso} = t_{accel} + t_{const} \text{ s}$$

Bisogna anche calcolare la **distanza totale** coperta durante il sorpasso, diversa da X_S in quanto nella seconda consideriamo velocità relative al *Platoon A* non le velocità effettive:

$$X_{total} = \bar{V}_{B\,accel} \cdot t_{accel} + V_B \cdot t_{const} \text{ m}$$

Dove $\bar{V}_{B\,accel} = \frac{V_B + V_{B0}}{2}$

Ora il dato fondamentale per decidere se effettura il sorpasso o valutare altre alternative, la **posizione finale** del *Platoon B*:

$$X_{Bf} = X_{B0} + X_{total} + \epsilon \text{ m}$$

Ricordiamo l'utilizzo di ϵ per aggiungere uno spazio di sicurezza alla fine del sorpasso.

Una volta ottenuti questi dati bisognerà calcolare la posizione finale del *Platoon C* dopo il tempo necessario al sorpasso:

$$X_{Cf} = X_{C0} - V_C \cdot t_{sorpasso} - \epsilon \text{ m}$$

3.3.1 Decisione manovra

Una volta calcolati tutti i dati necessari il leader del *Platoon B* valuterà la situazione in cui si trova e deciderà se effettuare la manovra:

1. $X_{Bf} < X_{Cf} - d_{head}$

La posizione finale del *Platoon B*, cioè la posizione in cui rientra nella propria carreggiata, è minore della posizione finale del *Platoon C* meno la distanza di sicurezza d_{head} . Questo vuol dire che il *Platoon B* non incrocerà il *Platoon C* durante il sorpasso. Il *Platoon B* **può effettuare** il sorpasso.

Nel caso in cui:

2. $X_{Bf} \geq X_{Cf} - d_{head}$

La posizione finale del *Platoon C* meno la distanza di sicurezza d_{head} è minore o uguale alla posizione finale del *Platoon B* vuol dire che i due platoon si incroceranno in un punto durante il sorpasso, causando un incidente. Il *Platoon B*, quindi, non può effettuare il sorpasso, possiamo valutare di far rallentare il *Platoon C* del 10% e valutare di nuovo la situazione:

$$X_{Cnew} = X_{C0} - (V_C \cdot 0.9) \cdot t_{sorpasso} - \epsilon \text{ m}$$

Se:

3. $X_{Bf} < X_{Cnew} - d_{head}$

Allora possiamo effettuare il sorpasso, altrimenti il *Platoon B* dovrà definitivamente attendere il passaggio del *Platoon C*.

Come detto in precedenza, si è scelto di modificare solamente la velocità del *Platoon C*, però se si modificasse la velocità anche degli altri platoon, aumentando quella del *Platoon B* e/o diminuendo quella del *Platoon A*, si potrebbe effettuare la manovra in più situazioni rispetto a quelle a cui siamo limitati ora.

3.3.2 Esempio numerico

Vediamo ora un esempio numerico per comprendere meglio l'algoritmo. Definiamo:

- $V_A = 50 \text{ km h}^{-1} = 13.9 \text{ m s}^{-1}$
- $V_B = 100 \text{ km h}^{-1} = 27.8 \text{ m s}^{-1}$
- $V_C = 50 \text{ km h}^{-1} = 13.9 \text{ m s}^{-1}$
- $V_{B0} = 27.8 \text{ m s}^{-1}$: il *Platoon B* non ha ancora rallentato a causa del CACC.
- $X_A = 150 \text{ m}$
- $X_{B0} = 50 \text{ m}$
- $X_{C0} = 420 \text{ m}$
- $\epsilon = 10$
- $d_{head} = 10$

Andiamo a calcolare i dati necessari per l'algoritmo decisionale:

$$d_{front} = \frac{(V_A \cdot 3.6) \cdot 3}{10} = 15 \text{ m}$$

$$X_S = X_A - X_{B0} + d_{front} = 150 - 50 + 15 = 115 \text{ m}$$

$$\Delta V_{BA} = (V_B - V_A) = 27.8 - 13.9 = 13.9 \text{ m s}^{-1}$$

Il calcolo può essere semplificato in quanto $V_{B0} = V_B$. Per lo stesso motivo possiamo evitare di calcolare t_{accel} e d_{accel} in quanto il platoon non dovrà seguire una fase di accelerazione per portarsi alla velocità desiderata. Avremo quindi $d_{rest} = X_S$

$$t_{const} = \frac{X_S}{V_B - V_A} = \frac{115}{27.8 - 13.9} = 8.3 \text{ s}$$

La distanza totale coperta sarà quindi:

$$X_{total} = V_B \cdot t_{const} = 27.8 \cdot 8.3 = 230.7 \text{ m}$$

E la posizione finale del *Platoon B*:

$$X_{Bf} = X_{B0} + X_{total} + \epsilon = 50 + 230.7 + 10 = 290.7 \text{ m}$$

Calcoliamo ora la posizione finale del *Platoon C*:

$$X_{Cf} = X_{C0} - V_C \cdot t_{sorpasso} - \epsilon = 420 - 13.9 \cdot 8.3 - 10 = 294.6 \text{ m}$$

Bisogna ricordare però che nella valutazione va aggiunta la distanza di sicurezza d_{head} , quindi:

$$X_{Bf} > X_{Cf} - d_{head} \Rightarrow 290.7 > 294.6 - 10$$

Valutiamo se diminuendo la velocità del *Platoon C* possiamo effettuare il sorpasso:

$$X_{Cnew} = X_{C0} - (V_C \cdot 0.9) \cdot t_{sorpasso} - \epsilon = 420 - (13.9 \cdot 0.9) \cdot 8.3 - 10 = 306.2 \text{ m}$$

$$X_{Bf} < X_{Cnew} - d_{head} \Rightarrow 290.7 < 306.2 - 10$$

Otteniamo che il *Platoon B* può effettuare il sorpasso solo comunicando al *Platoon C* di diminuire la propria velocità del 10%. Ci troviamo quindi nella situazione 3, descritta in precedenza, che dopo le nuove valutazioni ci porta nella situazione 1.

Da notare come molte distanze aggiunte per sicurezza possono essere diminuite o eliminate, ad esempio se non fossimo interessati ad avere uno spazio di sicurezza tra il *Platoon B* e il *Platoon C* potremmo eliminare d_{head} , in questo caso avremmo potuto eseguire subito la manovra in quanto:

$$X_{Bf} < X_{Cf} \Rightarrow 290.7 < 294.6$$

4 Simulazione e risultati

In questo capitolo in cui concludiamo la descrizione della manovra di sorpasso andiamo a vedere come viene impostata la simulazione, utilizzando i software e framework visti in precedenza; in seguito andremo a descrivere i diversi parametri utilizzati per effettuare una serie di simulazioni usate per dimostrare la fattibilità e l'efficacia della manovra di sorpasso. Infine, verranno presentati i risultati ottenuti da queste.

4.1 Setup simulazione

Vediamo come la simulazione è strutturata, dove si trovano e cosa fanno i file di interesse. La manovra di sorpasso è definita nella cartella `examples/` tra i file di Plexe, qui sono presenti i file di configurazione di OMNeT++ e SUMO, oltre ai file di output della simulazione.

4.1.1 SUMO

Tra i file di configurazione di SUMO troviamo `overtake.net.xml` responsabile per la definizione della rete stradale, come possiamo ad esempio vedere la carreggiata in Figura 3.1 definita in questo. Nel file `overtake.rou.xml` troviamo diverse informazioni tra cui i parametri e i modelli utilizzati per i veicoli; ad esempio sono definiti:

- **Accelerazione massima:** 1.5 m/s^2 nel nostro caso. Viene utilizzata nel calcolo dei tempi di sorpasso nell'algoritmo decisionale.
- **Lane-Changing Model:** definisce il comportamento a livello microscopico del veicolo quando sceglie di cambiare corsia [1]. In questo caso è stato utilizzato il modello LC2013.
- **Emission Class:** definisce la classe di emissione del veicolo, in questo caso è stata utilizzata la classe HBEFA3/PC_D_EU6. HBEFA3 si riferisce al database 'Handbook Emission Factors for Road Transport' [2] che fornisce i fattori di emissione per i veicoli stradali. PC_D_EU6 si riferisce al tipo di veicolo, un'auto privata diesel Euro 6.

È SUMO stesso a fornire tutti i dati sulle emissioni dei veicoli nei file `.vec` per ogni simulazione. In questi troviamo le emissioni di CO_2 ad ogni step di simulazione per ogni veicolo. La durata di uno step è modificabile, però in questo caso è stata mantenuta quella di default a 0.01 secondi.

4.1.2 OMNeT++

OMNeT++ ci mette a disposizione un file di configurazione `omnetpp.ini` in cui andremo a definire i parametri della simulazione, tra cui il modello dei veicoli, le specifiche per la comunicazione wireless, altri file di configurazione da utilizzare, dati per le nostre auto e i file da eseguire per la simulazione. Avere tutti questi dati definiti in un unico file ci permette di gestirli e modificarli con semplicità. Ad esempio se volessimo cambiare protocollo di Cruise Control utilizzato dai nostri veicoli basterebbe semplicemente cambiare il parametro `*.traffic.controller = "CACC"` che troviamo in questo file.

Tra i file da eseguire per la simulazione troviamo:

- **Traffic:** serve per specificare i dati iniziali dei veicoli, ad esempio dove sono posizionati all'avvio della simulazione, la velocità iniziale e il controllore di guida.
- **Scenario:** definisce come i veicoli si devono comportare nella simulazione.
- **Application:** l'obiettivo di questo file è di smistare e passare i dati ricevuti via wireless ai controller automatici.

- **Protocol:** il tipo di protocollo usato per la comunicazione wireless. Nel nostro caso è stato utilizzato il protocollo *SimplePlatooningBeaconing*, sviluppato nel framework Plexe. Questa classe implementa un classico protocollo di beaconing periodico inviando un beacon ogni x millisecondi.

Tutti i parametri definiti sono accessibili dai file NED che accompagnano i precedenti file. Il modulo *network* è definito nella cartella principale dal file `Overtake.ned`.

4.1.3 Plexe

Ora andiamo a vedere il percorso che prende il flusso del programma durante la simulazione. Spieghiamo come interagiscono i file in Figura 4.1, in che ordine vengono eseguiti e soprattutto che messaggi vengono scambiati.

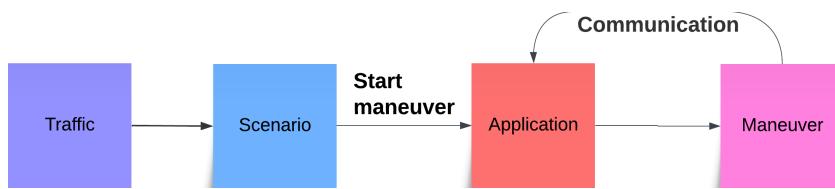


Figura 4.1: Flowchart della simulazione.

Traffic

Questo modulo è responsabile per la creazione dei veicoli e per la loro inizializzazione. Nel suo file NED oltre ad altri dati necessari per la simulazione sono definite le posizioni, velocità di partenza e il numero di veicoli nei platoon.

```

//platoon insertion speed
double platoonASpeed @unit("kmph");
double platoonBSpeed @unit("kmph");
double platoonCSpeed @unit("kmph");

//platoon size A -> overtaking
int platoonSizeA;
//platoon size B -> overtaker
int platoonSizeB;
//platoon size C -> opposite direction
int platoonSizeC;

int initialPositionDeltaA;
int initialPositionDeltaB;
int initialPositionDeltaC;
  
```

Figura 4.2: Parte di file NED del modulo traffic.

La classe `Traffic` assegna a tutti i veicoli i parametri necessari, ad esempio il platoon a cui appartengono e la loro corsia di partenza, per poi inserirli nella simulazione SUMO utilizzando le API a `Traci` messe a disposizione da Plexe.

Scenario

Una volta inseriti tutti i veicoli nella simulazione il modulo `Scenario` utilizza le API Plexe per assegnare il controllore e le velocità specificate nel file `omnetpp.ini` ai veicoli. È necessario utilizzare le API Plexe per trasmettere queste informazioni a SUMO. Nel nostro caso il controllore definito è un `CACC`. Inoltre questo modulo assegna ai diversi veicoli il proprio ruolo nel platoon, a quelli in testa viene assegnato `LEADER` mentre agli altri `FOLLOWER`.

Una volta terminato questo processo lo Scenario fa avviare al leader del *Platoon B* la manovra di sorpasso nel file `Application`.

Application

La prima azione del modulo `Application` è chiamare il metodo `startManeuver` nella classe `OvertakeManeuver`, questo metodo ha bisogno di sapere i dati di posizione, velocità e grandezza dei *Platoon A* e *Platoon C*. Il leader del *Platoon B* invia messaggi per richiedere i dati. Tutto questo meccanismo di scambio messaggi viene gestito dal modulo `Application`.

La possibilità di scambiare messaggi tra moduli viene offerta da OMNeT++ e dalle API Plexe. Vediamo il `ManeuverMessage` come esempio, i dati in questo verranno poi ereditati da tutti gli altri tipi di messaggio definiti per la manovra i quali, a loro volta, aggiungeranno dati specifici necessari ai loro scopi

```
// General message for an arbitrary maneuver to hold common information.
// Only children of this message should be initialized.
packet ManeuverMessage {
    // id of the originator of this message
    int vehicleId;
    // id of the platoon this message is about
    int platoonId;
    // id of the destination of this message
    int destinationId;
    // sumo external id of the sender
    string externalId;
}
```

Figura 4.3: `ManeuverMessage`.

Questi messaggio vengono definiti in un file di tipo `*.msg`, OMNeT++ li trasformerà in automatico in classi C++ con un nome simile: `*_m.h/cc`, che potranno essere utilizzati facilmente dai nostri moduli grazie ai classici metodi getter e setter dei parametri.

Il leader del *Platoon B* andrà a creare due messaggi `requestDataMessage` che saranno inviati ai leader di A e C; riceverà in risposta messaggi `VehicleDataMessage` con velocità, posizione e grandezza attuali del platoon corrispondente.

Altri messaggi gestiti da `Application` sono:

- `OvertakeMessage`: questo viene inviato dal leader di B ai suoi follower per informarli dell'avvio della manovra di sorpasso.
- `NewSpeedMessage`: questo messaggio viene inviato al *Platoon C* quando si decide di fargli diminuire la velocità per permettere il sorpasso.

Maneuver

Il file `Maneuver` non è un modulo OMNeT++, ma una semplice classe C++ utilizzata per gestire l'algoritmo decisionale e richiedere l'invio di alcuni messaggi. Dopo aver ottenuto i dati dei platoon A e C, questa classe esegue l'algoritmo decisionale e gestisce le eventuali situazioni in cui ci possiamo ritrovare. Ad esempio se valutiamo di diminuire la velocità del *Platoon C* per permettere il sorpasso, verrà inviato un messaggio `NewSpeedMessage` al leader di C.

Questa classe è responsabile dell'utilizzo delle API Plexe attraverso `plexetraciVehicle` per far cambiare al *Platoon B* la corsia quando necessario.

Una volta passato il tempo calcolato dall'algoritmo per il sorpasso: $t_{sorpasso}$, attraverso il metodo `plexetraciVehicle->changeLaneRelative(-1, 0)` il leader di B farà rientrare il platoon nella corsia di partenza.

Infine qua vengono registrate le informazioni di `Overtake_start`: inizio manovra, `Overtake_stop`: conclusione manovra e `Overtake_method` che indica la situazione in cui si è trovato il *Platoon B*.

4.2 Parametri della simulazione

Abbiamo già visto alcuni parametri definiti nel file `omnetpp.ini`, quelli che più ci interessano si trovano nella sezione dedicata al **Traffic manager**, qui vengono definiti:

- **PlatooningVType**: il modello di veicolo che vogliamo utilizzare, definito nel file `SUMO overtake.rou.xml`.
- Headways: le distanze che i veicoli devono mantenere rispetto ai veicoli davanti. Importante perché va ad influire sul quando il *Platoon B* inizierà a rallentare prima del cambio corsia.
- **PlatoonXSpeed**: le velocità di partenza dei 3 platoon:
 - A. $50 \frac{km}{h}$
 - B. $100 \frac{km}{h}$
 - C. Nelle simulazioni la velocità varia da 45 a $65 \frac{km}{h}$ con uno step di $1 \frac{km}{h}$
- **PlatoonSizeX**: il numero di veicoli nei platoon: sono stati considerati 2 veicoli per il **Platoon A** e 1 veicolo per il **Platoon B** e **Platoon C**.
- **InitialPositionDeltaX**: la posizione iniziale dei platoon: **A** parte da 150 m, **B** da 50 m e la posizione del **Platoon C** varia da 530 a 490 m con uno step di 2 m
- **Controller**: il Cruise Control utilizzato dai veicoli, CACC nel nostro caso.

4.3 Risultati

Definita la struttura della simulazione andiamo a vedere uno studio fatto partendo dai parametri elencati prima.

La velocità e la posizione del **Platoon C** variano su 21 valori ciascuno, creando quindi 441 simulazioni differenti.

Le simulazioni vengono avviate tramite il comando:

```
plexerun -u Cmdenv -c OvertakeNoGui -r i
```

Dove 'i' indica il numero seriale della simulazione. I numeri seriali vengono assegnati automaticamente da OMNeT++ grazie alla definizione di controlloer e delle variabili controllate da esso.

Andremo ad eseguire queste simulazioni 2 volte, la prima volta sarà possibile far rallentare il *Platoon C* per permettere il sorpasso, mentre la seconda volta rimuoveremo questa opzione, lasciando solo i casi 1 e 2 nell'algoritmo decisionale. Nelle seguenti sezioni andremo a confrontare i risultati delle due situazioni.

I risultati delle simulazioni si trovano nella cartella `results/`. I file che ci interessano sono: `OvertakeNoGui_{velocità}_{posizione}_0.vec`: da questo, grazie ad uno script messo a disposizione da Veins, possiamo ottenere i dati sulle emissioni dei veicoli in formato CSV. `OvertakeNoGui_{velocità}_{posizione}_0.sca`: qua troviamo i dati relativi a **Overtake_start**, **Overtake_stop** e **Overtake_method**. {velocità} e {posizione} sono i valori di velocità e posizione del **Platoon C** nella simulazione a cui appartengono i file.

4.3.1 Emissioni

Analizziamo le emissioni emesse nelle simulazioni. Dopo aver estratto i dati dai file `.vec` e averli trasformati in CSV possiamo andare a valutarli. SUMO ci fornisce le emissioni di tutti i veicoli ogni 0.01 secondi; possiamo ignorare i dati dei veicoli del *Platoon A* in quanto la loro velocità e la loro posizione di partenza rimane costante in tutte le simulazioni. Bisognerà invece sommare tutti i consumi dei veicoli del **Platoon B** e **Platoon C**.

In Figura 4.4 vediamo in **blu** le emissioni nelle simulazioni in cui il **Platoon C** non può rallentare e in **rosso** quelle in cui in certi casi rallenta.

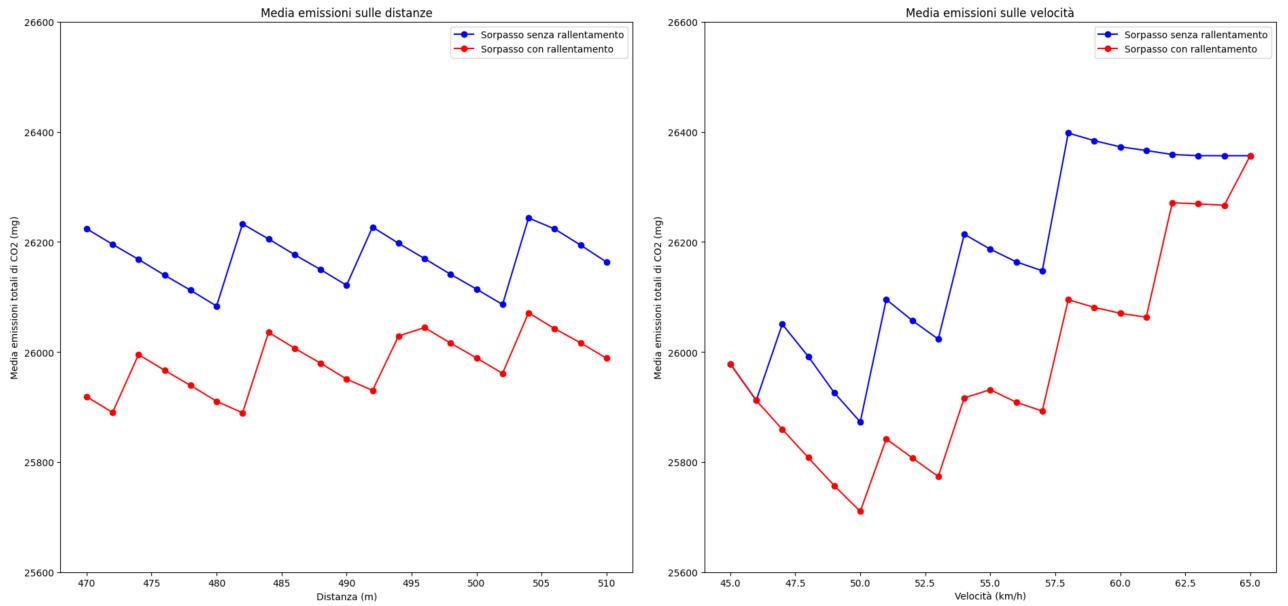


Figura 4.4: (Sinistra) Emissioni medie aggregate per distanza di partenza del **Platoon C**
 (Destra) Emissioni medie aggregate per velocità di partenza del **Platoon C**.

Tra tutte le simulazioni abbiamo una media di emissioni tra i 25.7 e 26.4 grammi di CO_2 . Come possiamo vedere, il rallentamento del **Platoon C** porta ad una diminuzione delle emissioni in quasi tutte le situazioni, questo è dovuto al fatto che in quei casi il **Platoon B** può effettuare il sorpasso in modo più efficiente, evitando di dover accelerare e decelerare più volte.

L'andamento a zig-zag dei grafici viene spiegato in dettagli nel paragrafo dedicato all'analisi dei casi sotto cui si esegue il sorpasso.

4.3.2 Tempo di sorpasso

Un altro dato interessante da considerare è quanto tempo impiega il **Platoon B** per completare la manovra:

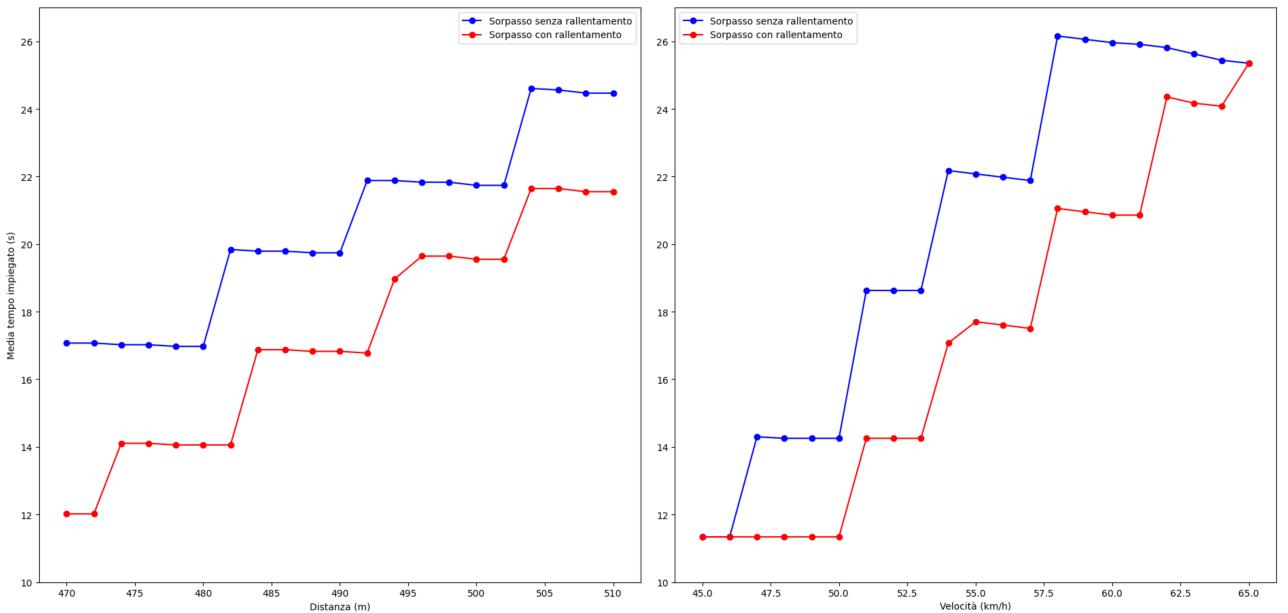


Figura 4.5: Tempi medi di completamento manovra per (Sinistra) distanza di partenza e (Destra) velocità di partenza del **Platoon C**.

Come vediamo anche in questa figura la manovra che prevede il rallentamento del **Platoon C** porta ad una generale diminuzione del tempo di sorpasso. Questo è principalmente dovuto al fatto che nei casi in cui *C* rallenta il **Platoon B** anzichè doverlo aspettare può iniziare la manovra prima, portandolo a concludere il sorpasso in anticipo.

4.3.3 Analisi dei casi

Infine possiamo andare a valutare in quali situazioni si è trovato il *Platoon B* a fine esecuzione dell'algoritmo decisionale e possiamo vedere quante volte è stato necessario far rallentare il **Platoon C** per permettere al **Platoon B** di anticipare il sorpasso:

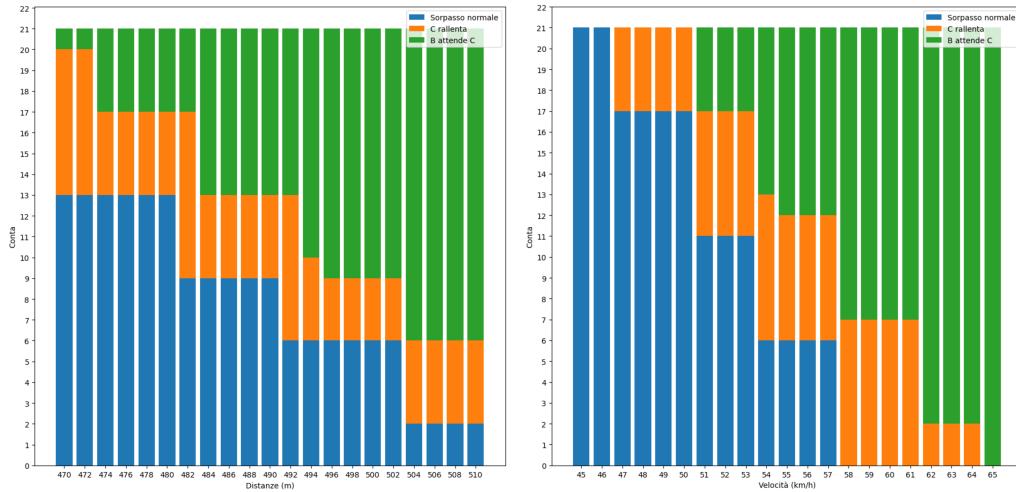


Figura 4.6: Scelta presa dall'algoritmo decisionale per la manovra con rallentamento del **Platoon C** a disposizione.

Vediamo in **blu** i casi in cui il *Platoon C* era abbastanza distante da permettere al **Platoon B** di effettuare il sorpasso senza rallentare, in **arancione** i casi in cui il **Platoon C** è stato fatto rallentare e in **verde** i casi in cui il *Platoon B* ha dovuto attendere il passaggio del *Platoon C*.

Le barre **arancioni** nelle simulazioni in cui è stata rimossa la possibilità di far rallentare il *Platoon C* diventano **verdi**; questa differenza spiega i minori consumi e tempi di sorpasso nei grafici visti in precedenza.

Da notare anche l'andamento a scala dei casi di sorpasso nei grafici, più la posizione di partenza del *Platoon C* è vicina al *Platoon B* e più è alta la velocità del primo, meno volte il *Platoon B* è stato in grado di eseguire il sorpasso senza attendere il *Platoon C*; questo spiega l'andamento a zig-zag dei grafici di emissioni e tempi di sorpasso.

4.3.4 Limiti del contributo

La manovra è stata sviluppata considerando sempre un **Platoon A** di lunghezza variabile e **Platoon B** e **Platoon C** composti da un solo veicolo. La maggior parte del codice è predisposto a gestire i casi in cui tutti i platoon sono di lunchezza variabile, il modulo **Traffic** può inserire un qualsiasi numero di veicoli nella simulazione, il modulo **Scenario** li inizializza correttamente e il modulo **Application** ha i metodi necessari alla comunicazione, tra cui la possibilità di avvisare tutti i membri di un platoon dell'avvio e della conclusione della manovra.

Le mancanze principali si trovano nel codice di **Maneuver** e nell'algoritmo decisionale. **Maneuver** non è ancora predisposto per avvisare i follower del leader del **Platoon B** di avviare o concludere il sorpasso e i messaggi necessari non sono stati creati. Invece l'algoritmo nei calcoli considera esplicitamente un solo veicolo per il **Platoon B** quando valuta la distanza per il sorpasso del **Platoon A** e la posizione del rientro del **Platoon B**, necessaria per evitare incidenti con il platoon sopraggiungente.

Un'altra limitazione della manovra allo stato attuale è l'assunzione che il **Platoon A** mantenga sempre la stessa velocità, è un'assunzione valida se i veicoli che lo compongono sono a guida autonoma oppure

se partecipano alla guida cooperativa, in quel caso potrebbe essere richiesto loro di mantenere la velocità durante il sorpasso, però nel caso più generale possibile questo non si può assumere.

5 Conclusioni

Le manovre cooperative senza ombra di dubbio portano un vantaggio concreto alla circolazione stradale. Oltre all'esempio del Platooning, anche il contributo principale della tesi ci dimostra come semplicemente aumentare la percezione dell'ambiente ai veicoli, ricevendo nel nostro esempio informazioni precise riguardo posizione e velocità degli altri veicoli, permetta una più efficiente gestione delle manovre svolte. Si è visto come alcune distanze di sicurezza possano essere ridotte permettendo lo svolgimento della manovra anche in casi in cui un guidatore umano o un'auto non cooperativa avrebbe dovuto rimandarne lo svolgimento.

Potenziando ancora di più le capacità cooperative, nel nostro caso richiedendo ad altri veicoli di svolgere azioni diverse da quelle che avevano previsto, ad esempio rallentare per permettere il passaggio di altri veicoli, porta ad ulteriori vantaggi; come dimostrato si ha una diminuzione dei consumi e dei tempi necessari alla manovra in praticamente tutte le situazioni.

5.1 Limitazioni

Nel capitolo precedente sono state esposte alcune limitazioni pratiche della manovra di sorpasso, queste possono essere superate proseguendo con lo sviluppo del progetto; in particolare per l'algoritmo decisionale basta estendere la formula per il calcolo dello spazio necessario al sorpasso aggiungendo la lunghezza del **Platoon B**:

$$X_S = X_A - X_{B0} + ((|B| \cdot L_{auto}) + ((|B| - 1) \cdot L_{headway})) + d_{front}$$

Dove $|B|$ è il numero di veicoli nel **Platoon B**, L_{auto} è la lunghezza di un veicolo e $L_{headway}$ è la distanza tra due veicoli.

E aggiustare i tempi di rientro in corsia per ogni veicolo del platoon.

Se ci concentriamo invece sulle limitazioni pratiche della guida cooperativa, spunta subito come principale ostacolo la diffusione dei veicoli in grado di usufruirne. Oltre alcune manovre specifiche, come il platooning, che non anno bisogno di auto a guida completamente autonoma per funzionare, molte altre la richiederanno e, nonostante ora stiano prendendo sempre più piede, una diffusione su larga scala di veicoli a guida autonoma è ancora lontana.

Anche se questa diffusione dovesse avvenire, rimarrebbe il problema della cooperazione tra veicoli di marche diverse, che potrebbero avere implementazioni diverse del protocollo di comunicazione e quindi non essere in grado di comunicare tra loro.

5.2 Lavori futuri

Nel futuro il principale contributo che si potrebbe portare al progetto sarebbe renderlo il più generale possibile, quindi aggiungendo la possibilità di avere macchine con diversi modelli e con diversi protocolli di funzionamento. Si potrebbe anche esplorare di quanto sia possibile ridurre le distanze di sicurezza, ad esempio d_{head} e d_{front} , tra i veicoli, permettendo così una maggiore efficienza della manovra. Sarebbe anche possibile sviluppare situazioni in cui il **Platoon C** non è disposto a collaborare, in quanto decide di eseguire azioni egoistiche. Questa situazione richiederebbe l'implementazione di messaggi di richiesta e conferma prima di richiedere la diminuzione della velocità del **Platoon C**.

Il contributo più interessante, però, sarebbe utilizzare queste simulazioni insieme a dati reali per allenare un algoritmo di intelligenza artificiale che possa prendere decisioni anche su situazioni più complesse e non prevedibili a priori, come quelle che abbiamo analizzato in questa tesi.

Infine l'obiettivo più importante è quello di riuscire a portare queste simulazioni fuori dal mondo virtuale e testarle su strada, passando prima a macchinine da laboratorio, poi ad ambienti reali protetti

e chiudendo su strade cittadine, per vedere se i risultati ottenuti in simulazione sono applicabili in situazioni reali.

Bibliografia

- [1] Jakob Erdmann. Lane-changing model in sumo. volume 24, 05 2014.
- [2] INFRAS. Handbook emission factors for road transport. <https://www.hbefa.net/> [Accessed: (2024)].
- [3] Daniel Krajzewicz, Georg Hertkorn, Christian Rössel, and Peter Wagner. Sumo (simulation of urban mobility) - an open-source traffic simulation. 2002.
- [4] Veronika Lesch, Martin Breitbach, Michele Segata, Christian Becker, Samuel Kounev, and Christian Krupitzer. An overview on approaches for coordination of platoons. *IEEE Transactions on Intelligent Transportation Systems*, 23(8):10049–10065, 2022.
- [5] Renato Lo Cigno and Michele Segata. Cooperative driving: A comprehensive perspective, the role of communications, and its potential development. *Elsevier Computer Communications*, 193:82–93, 9 2022.
- [6] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018.
- [7] Sumbal Malik, Manzoor Ahmed Khan, and Hesham El-Sayed. Collaborative autonomous driving—a survey of solution approaches and future challenges. *Sensors*, 21(11), 2021.
- [8] OMNeT++. *OMNeT++ Simulation Manual*.
- [9] Michele Segata. Vehicular networks. Lezione universitaria, 2024.
- [10] Michele Segata, Stefan Joerer, Bastian Bloessl, Christoph Sommer, Falko Dressler, and Renato Lo Cigno. PLEXE: A platooning extension for veins. In *2014 IEEE Vehicular Networking Conference (VNC)*, pages 53–60, 2014.
- [11] Michele Segata, Renato Lo Cigno, Tobias Hardes, Julian Heinovski, Max Schettler, Bastian Bloessl, Christoph Sommer, and Falko Dressler. Multi-Technology Cooperative Driving: An Analysis Based on PLEXE. *IEEE Transactions on Mobile Computing*, 22(8):4792–4806, 8 2023.
- [12] Christoph Sommer, David Eckhoff, Alexander Brummer, Dominik S. Buse, Florian Hagenauer, Stefan Joerer, and Michele Segata. *Veins: The Open Source Vehicular Network Simulation Framework*, pages 215–252. Springer International Publishing, Cham, 2019.
- [13] Christoph Sommer, Reinhard German, and Falko Dressler. Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis. *IEEE Transactions on Mobile Computing*, 10(1):3–15, January 2011.
- [14] András Varga and Rudolf Hornig. An overview of the omnet++ simulation environment. In *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, Simutools '08, Brussels, BEL, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).