UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Dipartimento di Scienze Fisiche,
Informatiche e Matematiche
Corso di Laurea Triennale in Informatica

Gestione degli incroci in una smart city: sistemi ad asta per veicoli autonomi e di emergenza

Relatori:
Prof. Giacomo Cabri
Prof. Manuela Montangero

Candidato:
Matteo Lugli
matr. 152590

Anno Accademico 2022-2023

Indice

1 Introduzione

La guida autonoma diventerà progressivamente parte integrante della nostra vita, sostituendo una delle attività umane tra le più stressanti e pericolose. Soltanto nel 2022 in Italia sono stati registrati più di 165 mila incidenti stradali che hanno provocato circa 3000 morti e 220 mila feriti [7]. Il ruolo principale della tecnologia è quello di creare soluzioni che mettano l'esperienza dell'uomo al centro, cercando di facilitare, migliorare e rendere più sicure le attività quotidiane. A tal proposito la ricerca si è evoluta molto rapidamente negli ultimi anni, proponendosi di trasformare radicalmente il modo in cui guidiamo. La Society of Automotive Engineers (SAE) descrive il coinvolgimento umano alla guida tramite una classifica basata su 5 livelli [8]:

- Livello 0
  Nessuna automazione: le auto non presentano dispositivi automatici in grado di dare assistenza attiva al guidatore, che deve sempre mantenere il pieno controllo del veicolo. Sono ricondotte a questo livello anche automobili equipaggiate con sistemi di frenata di emergenza o di rilevazione di collisioni imminenti.
- Livello 1
  Assistenza alla guida: il veicolo è dotato di sistemi come l'assistenza al mantenimento della corsia (LKA) o il controllo di velocità adattivo (ACC). Il conducente è comumque pienamente responsabile della guida, mantenendo il controllo del veicolo.
- Livello 2
  Automazione parziale alla guida: i veicoli sono equipaggiati con sistemi ADAS (*Advanced Driver Assistance Systems*) che in certe situazioni possono aiutare attivamente il conducente con il controllo dello sterzo, l'accellerazione e la frenata. *Autopilot* di *Tesla*, che ad oggi viene continuamente aggiornato tramite l'aggiunta di nuove funzionalità di assistenza alla guida, è ancora classificato come livello 2.
- Livello 3
  Automazione condizionata: il veicolo è in grado di gestire autonomamente la

guida in strada. Per legge, il conducente deve comunque essere sempre abilitato a poter mettere le mani sul volante e prendere il controllo dell'auto. Alcune compagnie come *Honda* hanno rilasciato sul mercato automobili di livello 3 considerabili completamente autonome in alcune zone urbane del giappone o sulle autostrade.

- Livello 4
  Automazione elevata: il veicolo è completamente autonomo nella guida e nella navigazione. Nonostante sia comunque necessaria la presenza di un conducente, il suo intervento non dovrebbe essere mai richiesto. Anche questi sistemi sono tipicamente funzionanti soltanto in certe aree geografiche.
- Livello 5
  Automazione completa: il veicolo è autonomo in qualsiasi condizione, e non è necessaria la presenza di un passeggero a bordo.

All'interno delle stesse compagnie automobilistiche e in ambito accademico, l'attenzione è prevalentemente rivolta al perfezionamento dei sistemi integrati nei singoli veicoli, guidando così la corsa verso il livello 5 di automazione. Tuttavia il traffico stradale è un sistema estremamente complesso, e tra non molti anni necessiteremo di tecnologie in grado di coordinare sistemi in cui interagiscono molti veicoli autonomi. L'obbiettivo principale sarà quello di sfruttare al meglio le risorse offerte dalle infrastrutture stradali, diminuendo i tempi di attesa nel traffico, i consumi di carburante o di elettricità e migliorare la sicurezza in strada. A tal proposito, gli incroci rappresentano nodi critici nella gestione del traffico: è infatti richiesto un delicato equilibrio tra l'assegnazione della precedenza in modo equo e la flessibilità necessaria per situazioni emergenziali. I metodi classici di assegnazione della precedenza agli incroci (semafori, segnali stradali, precedenza a destra, rotatorie) spesso non sono ottimali da questo punto di vista. È auspicabile che in un futuro prossimo, quando i veicoli equipaggiati con sistemi intelligenti saranno la maggior parte, essi vengano sostituiti con procedure che sfruttino al meglio le potenzialità delle nuove tecnologie.

Questa tesi si propone di studiare ed estendere alcuni algoritmi di coordinamento di veicoli autonomi agli incroci già presentati in letteratura. In particolare si parla di veicoli autonomi e connessi: ricadono cioè nel livello 5 di automazione e sono in

grado di comunicare tra di loro e con l'infrastruttura circostante. Nello specifico ci si è concentrati sugli algoritmi basati sulle aste, o *Auction Based*, in cui i veicoli sono in grado di gestire autonomamente una valuta digitale che gli permette di "acquistare" l'attraversamento dell'incrocio al bisogno. Un problema aperto per cui si è proposta una soluzione è quello dell'ottimizzazione dei tempi di attesa nel traffico dei veicoli d'emergenza. Si è discusso un metodo per far fluire il traffico in favore di questi ultimi senza condizionare in modo troppo invasivo i tempi di attesa degli altri veicoli. Inoltre è stata proposta una politica di gestione della valuta basata sul *deep reinforcement learning* che bilancia le spese e il tempo di attesa nel traffico.

Sono state svolte numerose simulazioni di traffico stradale per verificare che le proposte fossero effettivamente efficaci. I risultati degli esperimenti sui veicoli di emergenza mostrano la netta diminuzione dei loro tempi di attesa nel traffico, senza impattare in modo invasivo i tempi di attesa degli altri veicoli. Anche la strategia di gestione della valuta proposta si è rivelata efficiente: i tempi di attesa del veicolo su cui viene installato il gestore non si discostano di molto da quelli prodotti dalla strategia più comunemente usata in letteratura, mentre l'ammontare risparmiato in certi casi può arrivare al 70% rispetto al budget iniziale.

Nel capitolo 2 viene riassunta l'attuale situazione esistente, specificando in che contesto si è svolto il lavoro. Nel capitolo 3 si spiega su cosa verte la ricerca e qual'è stato il suo contributo innovativo. I dettagli implementativi vengono riportati nel capitolo 4. Nel capitolo 5 si descrivono nel dettaglio le condizioni in cui sono stati condotti gli esperimenti usando il simulatore di traffico, traendo poi le relative conclusioni nel capitolo 6.

2 Stato dell'arte

2.1 Introduzione e metodi di valutazione

Molti articoli scientifici si pongono in un ipotetico scenario in cui le strade sono popolate totalmente da veicoli a guida autonoma, in grado di comunicare tra di loro (*Vehicle to Vehicle comunication*, o V2V) e con dispositivi installati in prossimità degli incroci (*Vehicle to Infrastructure comunication* o V2I). Quando ciò sarà possibile, in condizioni di traffico scorrevoli i veicoli probabilmente non dovranno nemmeno fermarsi completamente agli incroci. Un gestore dotato di telecamere e sensori sarà in grado di comunicare al veicolo autonomo la velocità esatta con cui percorrere la sua traiettoria, senza scontrarsi con altri veicoli. Esistono algoritmi di coordinamento basati sulla prenotazione dello spazio necessario per percorrere la propria traiettoria per un lasso di tempo, anche chiamati *Reservation based*, appropriati per questo tipo di contesto [4]. Nonostante i progressi, si è ancora ben lontani dal vedere solo veicoli autonomi dominare le nostre strade. Il cambiamento dell'attuale paradigma di guida avverrà nel corso di molti anni, periodo in cui si dovranno gestire sia veicoli autonomi che veicoli non equipaggiati (massimo di livello 2). In tal caso i gestori degli incroci non potranno fare assunzioni sulla traiettoria dei veicoli, dato che questi potrebbero essere sia autonomi che non. Anche se i conducenti di veicoli non equipaggiati potessero comunicare la loro traiettoria agli altri veicoli o all'infrastruttura tramite un App installata sul loro smartphone o sul veicolo stesso, non si potrebbe in alcun caso presupporre che questa venga rispettata. Per tale motivo nel contesto di questa tesi si sono studiati algoritmi di coordinamento che non prendono in considerazione la traiettoria dei veicoli all'incrocio.

I metodi di valutazione principali usati in letteratura [1] per confrontare l'efficacia di diversi algoritmi sono i seguenti:

- tempo di attesa nel traffico, o *Traffic Waiting Time* (TWT): rappresenta il tempo in cui il veicolo è costretto ad aspettare in coda. Ovviamente si prediligono

sistemi che minimizzano questo parametro;

- tempo di attesa all'incrocio, o *Crossroad Waiting Time* (CWT): è il tempo speso dal veicolo in prossimità dell'incrocio, aspettando il suo turno di attraversamento. Anche in questo caso se ne vuole minimizzare il valore, per quanto possibile;
- bilanciamento del metodo, o *fairness*: si riferisce al principio di gestire tutti i veicoli nel modo più equo possibile.

2.2 Algoritmi *Auction Based*

Stone et.al [3] hanno proposto un algoritmo basato sulle aste, o *Auction Based(AB)*, in cui i veicoli che aspettano di oltrepassare l'incrocio effettuano una puntata in base alla loro disponibilità monetaria, per poi usare l'incrocio in ordine decrescente di puntata. Le caratteristiche, la distribuzione e l'origine della suddetta valuta non vengono indagate, così come i problemi di natura sociale ed economica che possono derivare dall'uso della stessa. In questo lavoro gli autori considerano la presenza di soli veicoli autonomi, anche se la gestione di veicoli non equipaggiati sarebbe idealmente possibile. In tal caso sarebbe richiesto:

- l'utilizzo di sensori appositi in grado di distinguere un veicolo non equipaggiato da un veicolo autonomo. Questi ultimi potrebbero identificarsi inviando un segnale di riconoscimento al manager dell'incrocio, che tramite un apposito dispositivo di ricezione potrebbe essere in grado di associare il segnale digitale all'entità fisica rilevata da moderni sistemi di visione (come telecamere o sensori lidar);
- il reimpiego degli impianti semaforici, che andrebbero riprogrammati per gestire nello specifico i veicoli non autonomi, comunicando tramite segnali luminosi quando il conducente ottiene il diritto di attraversamento.

Sotto queste ipotesi il manager dell'incrocio potrebbe associare ai veicoli non autonomi una puntata "standard" corrispondente ad un valore medio calcolato in base alle condizioni del traffico. I veicoli non equipaggiati sarebbero in tal modo trasparenti rispetto al sistema ad asta: il manager dovrebbe solo preoccuparsi di segnalare fisicamente il diritto di attraversamento al conducente al momento giusto.

Nel lavoro non vengono riportati dati relativi alla differenza di performance del sistema nell'utilizzo delle diverse tecniche di *bidding* proposte, elencate di seguito:

- *static*: gli agenti puntano sempre un'ammontare fissato di valuta, e hanno budget infinito;
- *free rider*: gli agenti puntano sempre zero;
- *fair*: gli agenti puntano ad ogni incrocio $\frac{found}{total}$, in cui $found$ rappresenta il budget attualmente a disposizione, mentre *total* indica il numero di incroci che il veicolo deve ancora attraversare per raggiungere la sua destinazione. Grazie a questo approccio l'agente non esaurisce mai il suo budget prima di terminare il suo percorso.

Sono state proposte molte varianti degli algoritmi bastati sulle aste, in cui cambia principalmente il modo in cui il manager concede l'attraversamento dell'incrocio ai veicoli partecipanti. Secondo gli esperimenti riportati in [1] la variante dell'algoritmo AB più performante in termini di tempi di attesa nel traffico e di fairness è quella cooperativa($COOP$), che prevede l'attraversamento di tutti i veicoli che hanno partecipato all'asta. In particolare, ogni veicolo effettua una puntata prelevando monete dal suo portafoglio virtuale, inviandola al manager dell'incrocio. Quest'ultimo raccoglie tutte le puntate dei veicoli partecipanti, per poi determinare l'ordine di attraversamento partendo dall'autore della puntata più alta, ossia il vincitore dell'asta. Al contrario, secondo il modello competitivo ($COMP$) è soltanto il vincitore ad attraversare l'incrocio: gli altri parteciperanno all'asta successiva.
Sono stati proposti diversi metodi per gestire il pagamento finale: secondo quello che si è rivelato essere più equilibrato è soltanto il vincitore a pagare la puntata (*only winner pays*).
La variante competitiva è particolarmente soggetta al rischio di *starvation*: se infatti un veicolo dovesse trovarsi ad un incrocio con poco budget a disposizione, potrebbe rimanere fermo per molto tempo, bloccando completamente la sua corsia. La tecnica che si è rivelata più efficace finora per aggirare il problema è la *bid redistribution*: la puntata vincitrice viene sempre re-distribuita equamente tra tutti i veicoli che hanno partecipato, che di conseguenza avranno più probabilità di vincere le aste successive.

In [1] l'*ehnancement* viene applicato per permettere anche alle auto che aspettano nel traffico di partecipare indirettamente all'asta. Per ogni auto che aspetta in coda dietro all'attuale veicolo che sta partecipando viene aggiunto un piccolo contributo positivo alla puntata di quest'ultimo, secondo la formula (1):

$$\begin{aligned} e(v_l) &= \log(n_l + 1) + 1 \\ b'_{v_l} &= e(v_l) * b_{v_l} \end{aligned} \tag{1}$$

in cui:

- v_l è il veicolo in testa alla corsia l;
- b_{v_l} è la puntata effettuata da v_l ;
- n_l è il numero di auto ferme nel traffico nella corsia l;

L'enhancement non prevede quindi che i veicoli nel traffico prelevino denaro dal loro wallet virtuale, semplicemente la loro presenza influenza in loro favore il flusso del traffico. In [2] Cabri et.al. introducono anche un meccanismo di *sponsorship* per permettere ai veicoli in attesa nel traffico di partecipare direttamente all'asta. Ogni veicolo può puntare una piccola percentuale del suo budget che verrà aggiunta alla puntata complessiva dell'attuale partecipante all'asta, ossia il veicolo in prossimità dell'incrocio.
Siano v_l il veicolo in testa alla corsia l e v_{l_i} il veicolo i -esimo in attesa nel traffico dietro a v_l. La puntata finale di v_l è pari a:

$$b'_{v_l} = b_{v_l} + \sum_i s_{v_{l_i}} \tag{2}$$

In cui

$$s_v = B * W_v \tag{3}$$

dove:

- B è un valore compreso tra 0 e 1;
- W_v è la disponibilità monetaria del veicolo v, ossia l'ammontare di valuta allocato per il percorso (prelevato dal wallet).

Ovviamente il meccanismo di sponsorship può essere abbinato all'enhancement, concatenando 1 e 2.
Nonostante questi due sistemi migliorino nel complesso i tempi di attesa e la fairness, permane il problema che nella variante cooperativa il sistema di aste determini soltanto l'ordine di attraversamento dei veicoli, senza considerare le situazioni di congestione del traffico. In uno scenario ideale, un algoritmo di coordinamento dovrebbe essere in grado di far fluire il traffico in base (ad esempio) alle esigenze dei veicoli di emergenza, per diminuire i loro tempi di attesa. Ciò significa che potrebbe essere necessario bloccare l'avanzamento di certe corsie per permettere a questi ultimi di di muoversi nel traffico più rapidamente, evitando le code agli incroci.

A tal proposito, in [2] non si sono notate differenze significative assegnando un budget molto alto (simulando una disponibilità di valuta infinita) ai veicoli di emergenza persino nella variante competitiva, principalmente per 3 motivi:

1. quando il veicolo vince l'asta effettuando una puntata molto alta, redistribuendo l'ammontare speso aumenta vertiginosamente l'inflazione: gli altri veicoli che hanno ricevuto una parte del denaro diventano a loro volta veicoli di emergenza, riducendo quindi il "potere d'acquisto" della valuta all'interno del sistema;
2. tramite il meccanismo di sponsorship, i veicoli di emergenza influsicono sul flusso del traffico soltanto nella corsia in cui sono, senza considerare le strade che dovranno percorrere successivamente;
3. i veicoli di emergenza in attesa nel traffico effettuano delle sponsorizzazioni altissime, causando il problema descritto nel punto 1 anche quando non sono in prossimità degli incroci.

2.3 Algoritmi adattivi

Negli ultimi anni si stanno integrando tecniche di *reinforcement learning* nella gestione delle infrastrutture stradali. Ad esempio, in [5] viene adottato un approccio dinamico al problema della gestione degli incroci, allenando un *manager* tramite il *Q-learning* [17] nello scegliere adattivamente il metodo migliore per far fluire il traffico in prossimità dell'incrocio ad ogni istante. In [16] Joo et.al. sfruttano il Q-learning per

allenare un gestore di semafori che cerca di massimizzare il throughput dell'incrocio minimizzando la deviazione standard delle lunghezze delle code. P. Karthikeyan et. al. [18] propongono un complesso sistema costituito da tre modelli (*Break safe control model*, *Intersection control model*, *Priority assignment model*) la cui continua interazione gestisce l'incrocio nel complesso. Il primo garantisce la sicurezza nel traffico evitando collisioni tra i veicoli, il secondo decide qual'è la politica di attraversamento migliore in ogni momento per massimizzare l'efficienza dell'incrocio mentre l'ultimo si occupa di gestire le precedenze dei veicoli che hanno inviato una richiesta al manager. Anche in questo caso l'algoritmo di reinforcement learning sfruttato nell'allenamento è il Q-learning.

2.4 Introduzione al Q-Learning

Il Q-learning è un algorimto di apprendimento frequentemente utilizzato nell'ambito del reinforcement learning (RL), principalmente per la sua efficienza e semplicità di implementazione. Il RL è una branca dell'intelligenza artificiale che si occupa dell'allenamento di agenti in grado di interagire con l'ambiente e ricevere segnali di feedback, in modo da ottimizzare il loro comportamento basandosi soltanto sulla loro esperienza. A differenza di altri metodi di *supervised learning* [13], il RL non presuppone l'uso di dati etichettati come esempi da fornire al modello. Gli elementi chiave del RL sono quindi i seguenti:

- *agente*: è l'entità che apprende e impara a prendere decisioni ottimali nell'ambiente in cui vive;
- *ambiente*: rappresenta l'insieme di fattori esterni che possono essere percepiti dall'agente, e con cui esso può interagire direttamente o indirettamente;
- *stato*: è una particolare istanza dell'ambiente in un certo istante. L'ambiente può avere un numero finito o infinito di stati, in base alla complessità del modello;
- *reward*: anche detto segnale di rinforzo, è il feedback che l'agente riceve dall'ambiente dopo aver eseguito un'azione che ne ha provocato il cambiamento.

L'idea che sta alla base del Q-learning è tutto sommato semplice: riempire una tabella (anche detta *Q-table*) che associ ogni stato s dell'ambiente con un'azione a che l'agente

deve eseguire per massimizzare l'*expected reward*, ossia il segnale di rinforzo che si aspetta di ricevere. Supponendo che gli stati dell'ambiente siano N e che le possibili azioni che può compiere l'agente siano M, allora la struttura della Q-table è quella riportata in Tabella 1. Ogni cella contiene il *Q-value* (o "*Quality value*") dell'eseguire l'azione a_j nello stato s_i.

Tabella 1: Q-table, N stati e M azioni.

	a_1	a_2	...	a_M
s_1	$Q(s_1, a_1)$	$Q(s_1, a_2)$	...	$Q(s_1, a_M)$
s_2	$Q(s_2, a_1)$	$Q(s_2, a_2)$	...	$Q(s_2, a_M)$
...	...	...	...	...
s_N	$Q(s_N, a_1)$	$Q(s_N, a_2)$	...	$Q(s_N, a_M)$

Quando l'agente si trova in un certo stato s_i, scegliendo $a_J | Q(s_i, a_J) = max_{a_j} Q(s_i, a_j)$ eseguirà l'azione che secondo la sua esperienza può massimizzare il segnale di rinforzo da quel momento in avanti. Questa scelta tiene conto sia della reward che ottiene immediatamente dopo l'esecuzione di a_J (anche detta *immediate reward*), sia delle reward che potranno ottenersi in futuro dallo stato s_{t+1} in cui l'agente è transitato eseguendo a_J nello stato s_i (*expected reward*).

La formula (4) descrive la regola di aggiornamento della Q-table nel tempo.

$$Q(s_t, a) := Q(s_t, a) + \eta(r + \gamma \cdot max_{a'} Q(s_{t+1}, a') - Q(s_t, a)) \tag{4}$$

In cui:

- η è il *learning rate*, cioè la velocità con cui si vuole far convergere il modello. E' ovviamente preferibile usare valori molto vicini a zero per ottenere una maggior precisione, a discapito della velocità di allenamento;

- γ è il *discount rate*, che indica lo sconto che viene applicato sull'expeted reward futura, cioè $max_{a'}Q(s_{t+1}, a')$. Più questo valore si avvicina a 1, più si da importanza a quest'ultima rispetto alla reward immediata. Sistemi estremamente complessi che puntano a predire molte azioni nel futuro (come il gioco degli Scacchi o del Go [20]) possono permettersi di usare valori molto alti;
- r è il valore della reward immediata, ottenuto eseguendo l'azione a nello stato s_t.

L'agente tramite un processo di tipo *trial and error* deve costantemente valutare il *tradeoff* tra *exploration* e *exploitation*. Deve cioè bilanciare l'esplorazione dello spazio delle azioni con lo sfruttamento dell'azione valutata come ottima in quel momento. Se si eccede con il primo si rischia di non convergere mai verso la policy ottima, emulando un comportamento pseudo-casuale. Se invece si predilige troppo la seconda si rischia di ricadere in dei minimi locali. Esistono diverse strategie la cui effiacia è stata dimostrata formalmente [17] per gestire il tradeoff. In realtà si usano più comunemente dei metodi ad hoc per cui non è stata dimostrata la convergenza che però si rivelano molto efficienti nella pratica.
Uno di questi consiste nel fissare un valore di probabilità ϵ per cui l'agente sceglie di svolgere un'azione casuale invece di sfruttare quella reputata migliore in quel momento (che invece è il comportamento di default).

2.5 Introduzione al Deep Q-Learning

Il Deep Q-Learning è una variante del Q-Learning che mira a risolvere il problema della cosiddetta *curse of dimensionality*. Infatti in certi casi è impossibile numerare tutti gli stati dell'ambiente, che potrebbe essere arbitrariamente complesso. Di conseguenza sarebbe impossibile costruire una Q-Table formata da un numero limitato di righe e colonne. Per questo nel Deep Q-Learning si usa una rete neurale al posto di una matrice (come mostrato in Figura 1) per associare ogni stato in input un vettore di Q-value. La funzione che se ne occupa è quindi memorizzata in modo distribuito nei pesi della rete, i quali vengono aggiornati gradualmente nella fase di training. Durante l'allenamento solitamente viene usata l'*experience replay* [14], una tecnica che consiste nel memorizzare e riutilizzare esperienze passate per migliorare l'apprendimento del

modello. Nella pratica si mantiene una struttura dati che possa memorizzare tuple del tipo $< s_t, a, r, s_{t+1} >$ in cui:

- s_t è lo stato dell'ambiente all'istante t;
- a è l'azione che ha eseguito l'agente nello stato s_t;
- r è la reward ottenuta dall'agente eseguendo l'azione a nello stato s_t;
- s_{t+1} è lo stato dell'ambiente all'istante $t + 1$.

Quando è il momento di aggiornare i pesi si estraggono casualmente dalla memoria B tuple su cui allenare il modello. In questo modo si riduce la correlazione temporale che sussiste tra i dati di addestramento, migliorando l'apprendimento. B è un iperparametro che prende il nome di *batch size*.

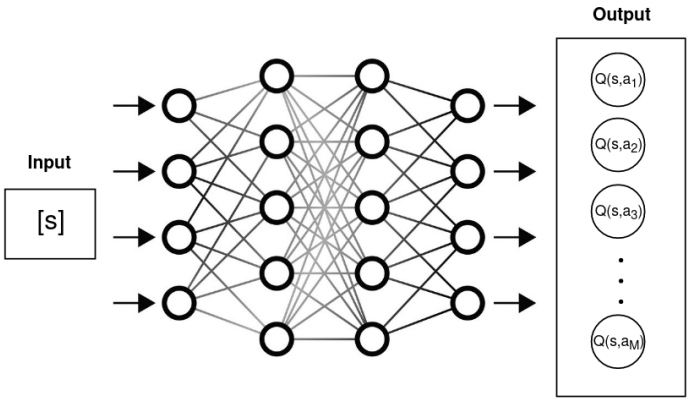

Figura 1: Deep Q-learning.

Per migliorare le performance durante l'allenamento del modello si usano in realtà due reti neurali con la stessa struttura ma con pesi diversi. La prima prende il nome di *Target Network*, ed è una sorta di "memoria episodica a lungo termine" con cui l'agente si confronta quando vuole capire come si sarebbe comportato in passato trovandosi in un certo stato s. La seconda prende il nome di *Q-Network*, e rappresenta la memoria a breve termine. Questa viene continuamente aggiornata ad ogni fase di training, e ogni K iterazioni viene copiata sostituendo il target network (essenzialmente la

memoria a breve termine diventa la nuova memoria a lungo termine). Anche K è un iperparametro che può essere soggetto a ottimizzazione.

Tendenzialmente nel *Deep Learning* [19] per migliorare le predizioni del modello si aggiornano i pesi della rete seguendo la regola della *backpropagation*. Nella pratica si modificano leggermente i pesi della rete in modo da minimizzare la *Loss Function (LF)*, ossia l'errore tra la predizione e l'output *target*. Solitamente si usa l'errore quadratico medio, ma la LF può variare in base al tipo di problema che si sta affrontando (classificazione, regressione, ecc ...). Nella formula (5) s_t, a, r, s_{t+1} sono i valori contenuti nella tupla che si è estratta dall'experience replay. $r + \gamma \cdot max_{a'} Q(s_{t+1}, a')$ viene calcolata usando il target network, mentre $Q(s_t, a)$ è la predizione calcolata usando il Q-network. É importante ricordare che la loss è una funzione dei parametri (θ) del modello.

$$L(\theta) = (r + \gamma \cdot max_{a'} \underbrace{Q(s_{t+1}, a')}_{\text{Target network}} - \overbrace{Q(s_t, a)}^{\text{Q-network}})^2 \tag{5}$$

3 Contributo della tesi

La tesi affronta la variante competitiva degli algoritmi basati sulle aste, che rispetto alla cooperativa offre molte più possibilità di analizzare l'impatto delle modifiche apportate alle strategie di bidding attuate da veicoli connessi.

Si osservi il caso riportato in Figura 2, in cui viene rappresentato graficamente un incrocio con alcune auto in attesa. Si supponga che tutti i veicoli abbiano la possibilità di effettuare delle sponsorship, e che il veicolo C (in rosso) abbia adottato una politica di gestione del budget che in questo incrocio massimizza la probabilità di vincere l'asta e che il traffico fluisca in suo favore. Secondo queste ipotesi, nella variante competitiva il veicolo C nel caso migliore avrebbe attraversato l'incrocio dopo $3t$ secondi, dove t è il tempo medio necessario che un veicolo impiega per attraversare un incrocio. Infatti l'ordine di attraversamento in questo caso sarebbe $(1, 5, C)$. Nella variante cooperativa invece il caso migliore prevede che C attraversi dopo $9t$ secondi: userebbero l'incrocio i veicoli $(1, 2, 3, 4, 5, 6, 7, 8)$ in qualsiasi ordine e solo successivamente attraverserebbe C.

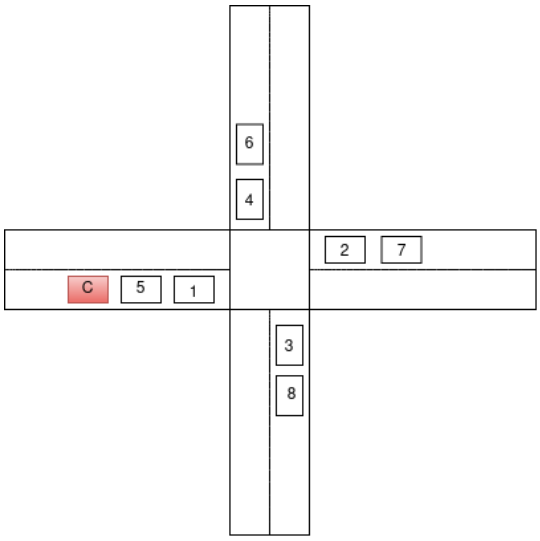

Figura 2: Incrocio in cui i rettangoli numerati rappresentano i veicoli, C è il veicolo campione.

L'esempio descritto spiega intuitivamente per quale motivo ci si è limitati allo studio della variante competitiva, che quindi sta alla base di entrambe le modifiche che sono state implementate. Si è pensato che soprattutto per lo studio di politiche innovative di gestione del budget fosse meglio basarsi su una variante che massimizzi i vantaggi apportati dall'uso di una strategia efficace.

Nella sezione 3.1 si parla ad alto livello della proposta riguardante i veicoli d'emergenza, mentre nella sezione 3.2 viene descritto il nuovo approccio *smart* alla gestione del budget dei veicoli.

3.1 Sound Boost per veicoli d'emergenza

È stata proposta una soluzione per ottimizzare la gestione dei veicoli di emergenza al fine di ridurre i loro tempi di attesa, mantenendo al contempo un impatto limitato sul traffico. Questa soluzione prende il nome di "Sound Boost" e si ispira al principio operativo dei veicoli di emergenza che utilizzano segnali sonori per avvisare gli altri conducenti. Il metodo emula un impulso che si propaga dalla posizione del veicolo di emergenza in modo esponenzialmente decrescente lungo la mappa, applicando un moltiplicatore alle puntate delle auto che si trovano su una strada che dovrà essere percorsa dal veicolo d'emergenza.

3.2 Gestore della valuta o *Bidder*

Durante lo studio degli algoritmi ad asta si è anche deciso di proporre un'innovativa strategia di bidding che potesse sostituire la classica strategia fair. Nello specifico è stato allenato un gestore delle puntate (o *bidder*) che impara a risparmiare più valuta possibile ma mantenendo dei buoni tempi di attesa nel traffico. All'inizio del percorso il proprietario del veicolo (se non ha urgenza di raggiungere in fretta la sua destinazione) potrebbe decidere di attivare il bidder in modo da risparmiare per le prossime corse. Il diagramma in Figura 3 mostra la logica di un menù di selezione che potrebbe essere installato sul sistema di *infotainment* dell'auto, grazie al quale il proprietario avrebbe la possibilità di decidere il budget da allocare per il percorso selezionato, insieme alla strategia di bidding più adatta per le sue esigenze.

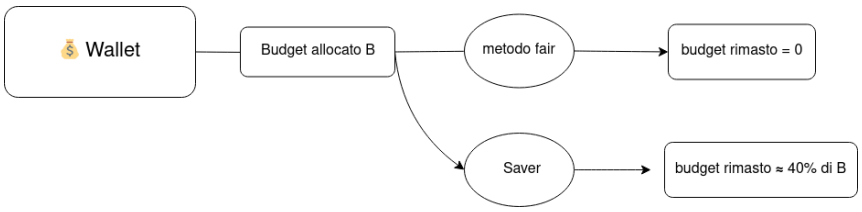

Figura 3: Logica del menù di selezione all'inizio del percorso

Il sistema integrato sul veicolo autonomo potrebbe inizialmenete comunicare al conducente la quantità di valuta che si aspetta di risparmiare sul percorso inserito in base a *(i)* la sua esperienza su quel percorso, *(ii)* l'esperienza di altri veicoli sullo stesso percorso, *(iii)* il budget allocato per quella specifica corsa, *(iv)* il traffico che si aspetta di incontrare.

Come anticipato il metodo proposto si basa sul *machine learning* (ML): fa uso dei dati accumulati nel tempo per imparare una strategia che ottimizzi il bilanciamento tra tempi di attesa e valuta spesa. In questa tesi si è considerata soltanto l'esperienza del singolo veicolo, che memorizza i dati di interesse in una memoria locale. In un lavoro futuro si potrebbe indagare il caso in cui tutti i veicoli che intraprendono spesso un determinato percorso P contribuiscano ad un *database* comune relativo a P con la loro esperienza. In questo modo il modello di ML verrebbe allenato su dati raccolti cooperativamente da molte auto, e ogni veicolo avrebbe già a disposizione una strategia di bidding valida per P senza averlo mai percorso prima. Un approccio simile sarebbe molto utile nel mondo reale, in cui i singoli veicoli faticherebbero nel raccogliere dati sufficienti contando singolarmente sulle loro forze. Per rendere il modello ancora più realistico, ogni database potrebbe essere ulteriormente diviso in più *subset* $s^{(p)}_{[t_1,t_2]}, s^{(p)}_{[t_2,t_3]}, \ldots, s^{(p)}_{[t_{n-1},t_n]}$, ognuno relativo ad una specifica fascia oraria. In questo modo sarebbe possibile allenare (quindi rendere usufruibile) un modello specifico per ogni intervallo di tempo $[t_i, t_{i+1}]$

Per quanto riguarda il processo di allenamento ci si è ispirati a quello che farebbe manualmente il conducente se le puntate non fossero automatizzate. Quando è il momento di prelevare denaro dal budget allocato per partecipare ad un'asta o per effettuare una sponsorship, si favoriscono scelte che hanno portato il veicolo a migliorare la sua posizione nel traffico in passato. Per emulare questo procedimento si è

sfruttato il paradigma del reinforcement learning, essendo il contesto particolarmente adatto. Infatti un veicolo che non ha mai intrapreso il percorso designato non ha esperienza a priori da cui estrapoloare le informazioni necessarie per attuare un'efficace strategia di bidding. L'agente quindi impara basandosi esclusivamente sulla sua esperienza passata, ottenendo una reward positiva quando tramite le sue scelte riesce a migliorare la sua posizione nel traffico. L'obbietivo finale dell'agente è quello di trovare una politica π che massimizzi la funzione di reward nel tempo, quindi che trovi un bilanciamento ottimo tra valuta risparmiata e tempi di attesa nel traffico. Nello specifico è stato implementato un *plugin* per il simulatore di riferimento che fa uso del Deep Q-Learning per allenare un veicolo campione.

4 Implementazione

Come simulatore è stato scelto SUMO (Simulator of Urban Mobility) [9] con cui ci si può interfacciare tramite TraCI (Traffic Control Interface) [10]. L'intera *codebase* utilizza Python 3.9 come linguaggio di programmazione. L'infrastruttura di controllo della simulazione è la stessa utilizzata in [6], a cui sono state apportate alcune modifiche(4.4, 4.5, 4.6).

4.1 Mappa

La mappa di riferimento è mostrata in Figura 4. E' una *Manhattan Grid* 5x5, che conta quindi un totale di 25 incroci. Tuttavia per semplicità sono stati presi in esame e gestiti tramite le aste soltanto i 9 incroci interni, mentre gli incroci esterni servono soltanto per la redirezione dei veicoli. E' stata scelta questa topologia per evitare

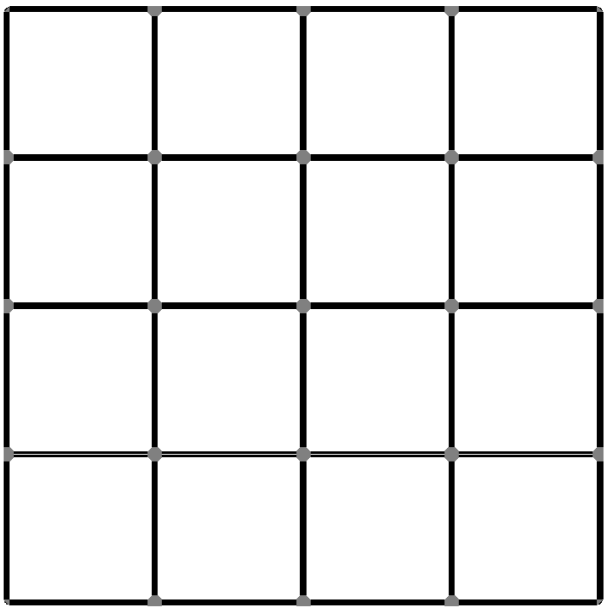

Figura 4: Manhattan grid 3x3

problemi tecnici dati dalla presenza di rotonde, sensi unici, attraversamenti pedonali, parcheggi e altre infrastrutture che non sono state oggetto di studio. Una mappa semplice e non troppo estesa ha permesso anche di supervisionare le simulazioni in

modo efficace quando necessario. Esistono strumenti come *Open Street Map* [11] adatti per convertire una zona geografica di interesse in un file compatibile con SuMO, ma si è iniziato da un'ambiente artificiale per aggirare i problemi elencati prima. Tuttavia l'uso di una mappa reale sarebbe sicuramente un argomento interessante da approfondire in un lavoro futuro.

4.2 Incroci

Gli incroci, o *junction*, sono in corrispondenza delle intersezioni tra le strade. Ognuna di queste è costituita da una sola corsia per senso di marcia. Come si può vedere in Figura 5, ad un incrocio possono aspettare al massimo 4 veicoli. Il numero in azzurro indica il budget attualmente a disposizione del veicolo, mentre il numero in bianco è l'identificativo del veicolo.

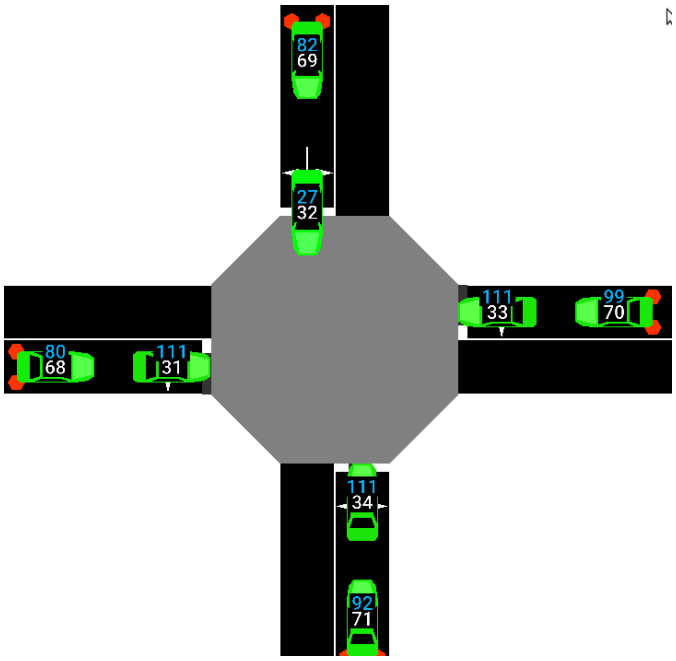

Figura 5: Esempio di incrocio

4.3 Veicoli

Tutti i veicoli simulati sono autonomi e connessi. I loro attributi fondamentali elencati di seguito:

- *Budget*: è un valore intero che descrive la disponibilità monetaria del singolo agente. Quando il veicolo viene generato, ossia all'inizio della simulazione, il budget viene solitamente impostato ad un valore che può variare in base al tipo di veicolo. Nel caso dei veicoli di emergenza (ad esempio) il budget iniziale è molto più alto rispetto ai veicoli "standard", i quali invece vengono inizializzati indistintamente con una valuta costante. Ulteriori dettagli saranno forniti nella sezione 4.5. Per le puntate agli incroci i veicoli adottano la strategia fair, mentre per le sponsorship seguono la formula (3). B viene scelto nell'intervallo $[0, \beta]$ in modo casuale. Essenzialmente ogni agente può sponsorizzare il veicolo in testa alla propria corsia al massimo del $\beta\%$ del proprio budget attuale. Il parametro β varia a seconda degli esperimenti.
- *Percorso*: all'inizio della simulazione ad ogni veicolo viene assegnato deterministicamente un percorso circolare. Ciò significa che quando un veicolo termina il suo percorso non viene eliminato dalla simulazione, ma inizia nuovamente dal punto di partenza. I vantaggi di questo metodo di *re-routing* sono i seguenti:
  - si possono analizzare i tempi di percorrenza del singolo veicolo sul lungo periodo, mentre variano le condizioni del traffico;
  - assegnando sempre lo stesso percorso ai veicoli si può studiare l'esperienza di guida di un certo veicolo A al variare di altri parametri (come il numero totale di veicoli sulla mappa,la durata della simulazione, il metodo di bidding adottato dai vari agenti, ecc...) essendo sicuri che il suo percorso non cambi tra una simulazione e l'altra. Questo comportamento simula la percorrenza di tragitti ricorrenti nella routine quotidiana, come il percorso "casa-lavoro-casa".

  Prima di ogni incrocio viene inserito uno *Stop*, necessario perchè il veicolo si fermi in prossimità dell'incrocio e non attraversi sfruttando la precedenza a

destra, ossia il comportamento di default. Il sistema di aste rimuoverà poi lo Stop al veicolo che si aggiudica l'attraversamento.

Per il calcolo dei tempi di attesa nel traffico e agli incroci il procedimento è analogo:

- Per il TWT viene misurato il tempo in cui il veicolo ha velocità prossima a zero ma non è nello stato di *stop* (in tal caso è fermo ad un incrocio);
- Per il CWT viene misurato il tempo in cui il veicolo rimane nello stato di *stop* agli incroci.

4.4 Risoluzione parallela delle aste

Il lavoro svolto in funzione di questa tesi ha previsto anche la risoluzione di alcune criticità presenti nel simulatore usato in [1], [2], [6] per svolgere gli esperimenti. L'obbiettivo principale è stato quello di rendere la simulazione più realistica implementando la risoluzione parallela delle aste, che prima venivano gestite sequenzialmente. L'algoritmo 1 descrive ad alto livello la modalità con cui si itera su gli incroci per risolvere le aste secondo il paradigma sequenziale.

Algoritmo 1 DepartCars, sequenziale

```
dc = dizionario contenente le auto in partenza da tutti gli incroci
C = lista degli incroci
for i in 1...4 do
    for c in C do
        if i <= len(dc[c]) then   ▷ assumendo che gli indici delle liste partano da 1
            depart(dc[c][i])
        end if
        traci.simulationStep()
    end for
end for
```

I due cicli `for` annidati servono per iterare su tutti i veicoli i -esimi nelle liste dei veicoli che hanno il diritto di attraversare l'incrocio sulla mappa. La funzione `departCars`

è infatti usata indistintamente per la variante cooperativa e per quella competitiva: nella prima ad ogni incrocio tutte le auto ottengono il diritto di attraversare, mentre nella seconda soltanto un veicolo usa l'incrocio. Ciò significa che nel COOP la lunghezza di `dc[c]` è al massimo 4, mentre nel COMP è al massimo 1.
La primitiva `depart(dc[c][i])` semplicemente rimuove lo stato di stop dal veicolo selezionato (quindi l'auto i -esima che ha il diritto di attraversare dall'incrocio c).
La funzione `traci.simulationStep()` invece fa avanzare di 1 passo tutti i veicoli nella mappa che non sono nello stato di stop.

Strutturare il codice in questo modo ha il vantaggio che per ogni i, alla fine del secondo ciclo annidato hanno attraversato tutti i veicoli i -esimi della mappa. In questo modo non vengono mai fatti partire simultaneamente due veicoli provenienti dallo stesso incrocio. Si ricordi infatti che il manager non può allocare porzioni di strada a due veicoli che attraversano insieme, non conoscendo le traiettorie degli stessi (si veda il capitolo 2). Tuttavia eseguendo un passo di simulazione [1] ad ogni iterazione del secondo ciclo `for`, il tempo avanza anche per tutti quei veicoli fermi agli incroci successivi il cui stato di stop dovrebbe essere rimosso alle prossime iterazioni. Questi veicoli rimangono quindi fermi ad aspettare il loro turno, quando in realtà potrebbero già utilizzare l'incrocio. L'algoritmo è stato quindi migliorato favorendo la risoluzione parallela di tutti gli incroci, effettuando un passo di simulazione una volta tolto lo stato di stop da tutti i veicoli i-esimi che hanno il diritto di attraversamento. Lo pseudocodice relativo al nuovo metodo è riportato in 2.

Questa modifica rende nel complesso la simulazione più lenta, perchè ad ogni chiamata di funzione vengono effettuati molti meno passi di simulazione, ma il traffico fluisce in modo più realistico.

[1] un passo di simulazione fa avanzare tutti i veicoli non nello stato di stop sulla mappa di 1 passo.

Algoritmo 2 DepartCars, parallelo

```
dc = dizionario contenente le auto in partenza da tutti gli incroci
C = lista degli incroci
for i in 1...4 do
    for c in C do
        if i <= len(dc[c]) then   ▷ assumendo che gli indici delle liste partano da 1
            depart(dc[c][i])
        end if
    end for
    traci.simulationStep()
end for
```

4.5 Veicoli di emergenza

Un veicolo di emergenza, o *EV*, è un veicolo a cui viene inizialmente assegnato un budget infinito. Ha quindi la possibilità di effettuare puntate e sponsorship molto più alte rispetto agli altri agenti. Questo gli permette di vincere sempre l'asta quando in prossimità di un incrocio e di far fluire il traffico sempre in suo favore quando aspetta in coda. Per risolvere i problemi descritti nel capitolo 2 sono state apportate alcune ulteriori modifiche:

- se il vincitore di un'asta è il veicolo di emergenza (quindi ad ogni asta a cui partecipa) la sua puntata vincente non viene re-distribuita tra gli altri partecipanti. Il meccanismo della redistribuzione della puntata è stato spiegato nel dettaglio nel capitolo 2.
- la percentuale della puntata finale di ogni veicolo dovuta alla sponsorship non viene più re-distribuita tra gli altri veicoli in attesa. In questo modo se un veicolo riceve una sponsorship da parte dell'EV, eviterà di redistribuire l'ammontare ricevuto aumentando l'inflazione nel sistema. Nella proposta viene quindi re-distribuita agli altri veicoli in attesa soltanto l'esatta quantità di denaro che il vincitore ha prelevato dal suo wallet.

Il contributo fondamentale del Sound Boost è quello di far fluire il traffico in favore

dell'EV considerando non solo la corsia in cui è attualmente, ma anche quelle che dovrà attraversare in futuro. Per questo è stato applicato un moltiplicatore alle puntate di tutti i veicoli che attualmente si trovano su una strada compresa nell'intero percorso dell'EV. Per evitare di agire in modo troppo invasivo sui tempi di attesa dei veicoli che si trovano molto lontani dall'EV, il sound boost diminuisce esponenzialmente all'aumentare della distanza tra il veicolo che lo riceve e l'EV.
In Figura 6 il veicolo azzurro è l'EV, da cui si propaga il segnale di potenziamento delineato in giallo. La strada blu denota il percorso che l'EV sta seguendo.
La formula del Sound Boost è definita nell'espressione (6):

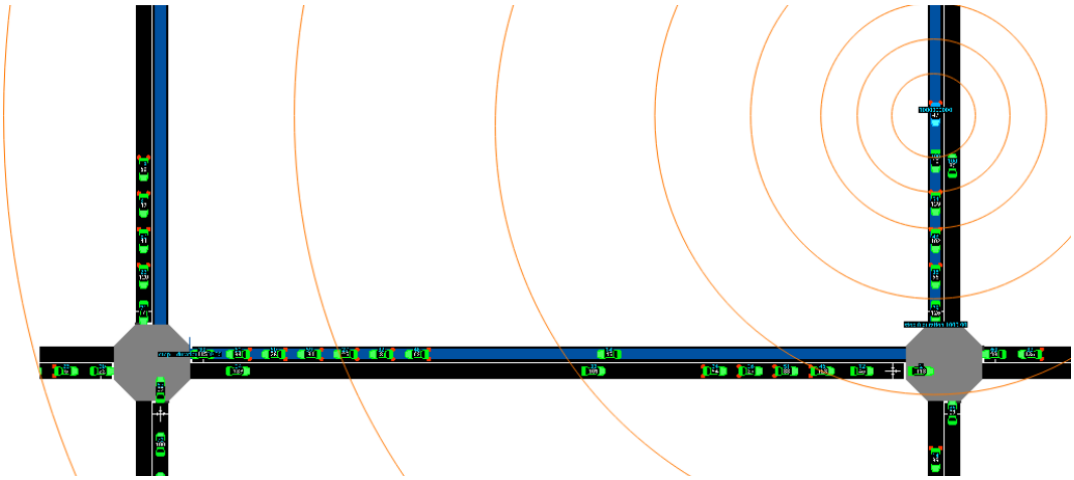

Figura 6: Ingradimento nelle vicinanze dell'EV

$$b'_v = b_v * (1 + e^{\frac{1}{\log \sqrt{D}}}) \tag{6}$$

in cui:

- b_v è l'attuale puntata del veicolo che sta ricevendo il boost;
- D rappresenta la distanza in linea d'aria tra il veicolo d'emergenza e il veicolo che riceve il boost.

D si trova al denominatore dell'esponente dato che il segnale di potenziamento deve diminuire all'aumentare della distanza e viceversa. Si usa la radice quadrata (funzione monotona crescente) per rallentare l'aumento di D. La formula proposta funziona bene

sulla mappa presa in considerazione: cambiando la mappa potrebbe essere necessario calibrare l'intensità del segnale applicando un moltiplicatore o aggiungendo un *bias* costante al valore di D. Ovviamente per non ricadere nel problema dell'inflazione, la percentuale di ogni puntata dovuta alla ricezione del Sound Boost non viene redistribuita.

4.6 **Bidder**

In questa sezione viene discussa ad alto livello l'implementazione del software adibito all'allenamento e al testing del gestore della valuta. Il diagramma in Figura 7 mostra le interazioni fondamentali tra i vari componenti che contribuiscono al funzionamento del bidder. Questi verranno descritti nel dettaglio nella sezione 4.6.1. La specifica struttura della rete neurale verrà discussa nella sezione 4.6.2, mentre nella sezione 4.6.3 vengono illustrate le differenze tra i due modelli che sono stati allenati. Nella sezione 4.6.4 sarà riportato lo pseudocodice dell'algoritmo di training, quindi dell'effettiva implementazione del Deep Q-learning. Nella sezione 4.6.5 invece viene commentato il codice di alcune primitive fondamentali.

4.6.1 Componenti

- *agente*: in questo caso si tratta del bidder installato sul veicolo, il cui obbiettivo è quello di imparare a risparmiare più valuta possibile su uno specifico percorso;
- *azione*: l'azione viene scelta dall'insieme finito di azioni $A = \{0, 0.1, 0.2, ..., 0.9, 1\}$ che rappresenta l'insieme di possibili sconti che possono essere applicati sulla puntata calcolata tramite il metodo fair. In dettaglio, le puntate sono calcolate usando la formula (7), in cui a rappresenta l'azione restituita dal bidder. Le sponsorship invece sono calcolate usando la formula (8), in cui T indica il numero di incroci totali che il veicolo deve attraversare. E' quindi un valore statico, a differenza di *total* (numero di incroci rimanenti che il veicolo deve attraversare, come spiegato nel capitolo 2). Se il percorso del veicolo prevede l'attraversamento di 10 incroci, il valore della sponsorship sarà sempre $\frac{found}{10}$.

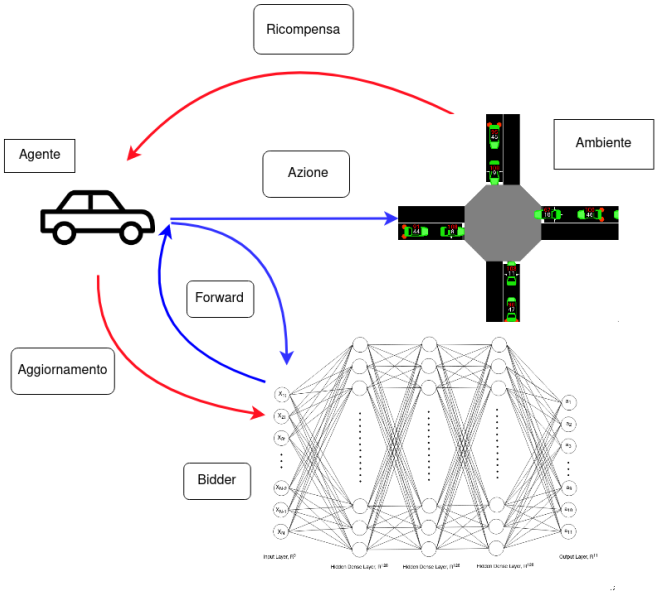

Figura 7: Allenamento del bidder. La freccia rossa indica il flusso del segnale di rinforzo. Esso viene ricevuto dall'agente che di conseguenza aggiorna la sua politica di comportamento. La freccia blu mostra come l'agente prima di agire chieda al bidder qual'è l'azione migliore in base allo stato attuale dell'ambiente.

$$b_v = \frac{found}{total} \cdot a \quad \text{con} \quad a \in A \tag{7}$$

$$s_v = \frac{found}{T} \cdot a \quad \text{con} \quad a \in A \tag{8}$$

- *stato*: lo stato s all'istante di tempo t è una rappresentazione compatta dell'ambiente che l'agente percepisce intorno a sè. Nello specifico si tratta di un vettore di N elementi, dove N è il numero totale di incroci sulla mappa:

$$s_t = \begin{bmatrix} x_{1t} \\ x_{2t} \\ \vdots \\ x_{Nt} \end{bmatrix} \tag{9}$$

in cui

$$x_{it} = \begin{cases} \frac{L-p}{L}, & \text{se il veicolo si trova all'incrocio } i \\ 0 & \text{altrimenti} \end{cases} \tag{10}$$

dove L rappresenta il numero totale di auto presenti nella corsia in cui si trova il veicolo attualmente, mentre p rappresenta la posizione del veicolo all'interno della coda stessa. p è un valore intero che varia nell'intervallo $[0, len]$.
Se $p = 0$, il veicolo si trova in prossimità dell'incrocio (in prima posizione), se $p = 1$ in seconda posizione, e così via. Questa codifica permette all'agente di capire per certo in quale incrocio si trova e approssimativamente qual'è la sua posizione nella coda di veicoli.

4.6.2 Struttura della rete

L'input della rete ad ogni iterazione corrisponde al vettore s_t. Seguono 3 *layer* densi, ognuno da 128 neuroni, per cui è stata usata la *relu* come funzione di attivazione. L'output è il vettore delle azioni: ogni neurone contiene il *Q-Value* della corrispettiva azione, ossia il valore che indica quanto è vantaggioso scegliere quella particolare azione dato lo stato in input. Per questo quando l'agente deve scegliere lo sconto da applicare alla sua bid sceglierà l'azione che ha Q-Value maggiore tra tutti i neuroni dell'output. In Figura 8 è mostrata la struttura della rete neurale, comune a entrambi i modelli allenati.
Se ad esempio l'output della rete (preso in input un generico vettore s) fosse $\begin{bmatrix} 0 & 0.3 & 0.4 & 0.2 & 0.1 & 0 & 0 & 0 & 0 \end{bmatrix}$ allora verrebbe restituita l'azione con indice uguale a 2 (a_2), che per comodità corrisponde ad uno sconto del 20% sulla puntata.

In figura 9 è riportato l'output del del metodo `summary()` fornito da Keras [12], che descrive in breve la struttura della rete neurale. Il Q-network e il target-network hanno la stessa struttura, quindi l'output è identico per le due reti.

4.6.3 Modelli

Nel contesto di questa tesi sono stati allenati due modelli, che sono stati poi valutati indipendentemente. Ogni modello si contraddistingue principalmente per la funzione di attivazione del layer di output e per la funzione di reward.

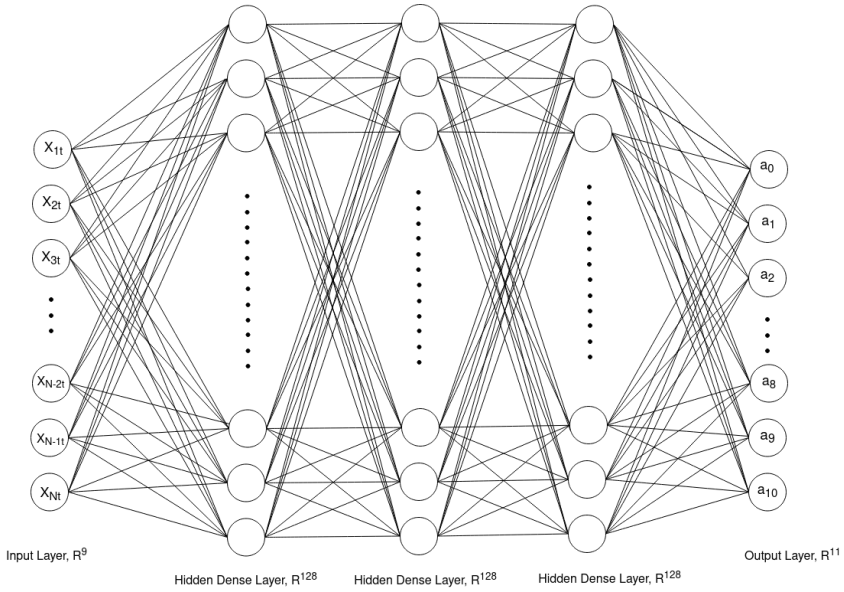

Figura 8: Rappresentazione della struttura interna del bidder, rete neurale

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 128)               1280

 dense_1 (Dense)             (None, 128)               16512

 dense_2 (Dense)             (None, 128)               16512

 dense_3 (Dense)             (None, 11)                1419

=================================================================
Total params: 35723 (139.54 KB)
Trainable params: 35723 (139.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figura 9: Sommario della struttura del Q network e del target network

1. Modello 'bidderV1'

   - *AF layer di output*: è stata usata la 'linear', che mantiene invariati i valori positivi e negativi;
   - *funzione di reward*: è una media pesata di due componenti. La prima ricompensa il veicolo se ha migliorato la sua posizione nel traffico, la seconda dipende direttamente dall'azione a, ossia dallo sconto applicato sulla puntata o sulla sponsorship. La reward aumenta all'aumentare dello sconto, quindi l'agente dovrebbe prediligere sconti alti. La formula (11) descrive in linguaggio matematico la funzione di reward.

$$
\begin{aligned}
R_p &= \begin{cases} c, & \text{se il veicolo vince l'asta} \\ p - P & \text{altrimenti} \end{cases} \\
R_d &= 1 - a \\
R_{tot} &= \alpha * R_p + (1 - \alpha) * R_d
\end{aligned} \tag{11}
$$

   Dove:

   - p è la posizione del veicolo in coda della coda nello stato precedente;
   - P è la posizione del veicolo in coda nello stato attuale. In altre parole, se il veicolo si è avvicinato all'incrocio $p - P$ vale 1, se è rimasto fermo vale 0. Se invece nell'ultima transizione di stato ha attraversato l'incrocio, allora R_p vale c;
   - c è una costante arbitraria, rappresenta la ricompensa che l'agente ottiene quando vince un'asta, quindi nel momento in cui attraversa l'incrocio. Dato che la reward per aver migliorato la sua posizione nel traffico vale al massimo 1, c è solitamente un valore maggiore di 1. In questo modo si vuole dare più importanza alla vincita di un'asta che alle singole sponsorship.
   - a è lo sconto restituito dal bidder;
   - α è un parametro utile per bilanciare l'importanza di migliorare la propria posizione nel traffico rispetto all'applicare degli sconti alti.

2. Modello 'bidderV2'

   - *AF layer di output*: 'softmax', che restituisce un vettore di probabilità preso in input un vettore di numeri reali. E' particolarmente utile nel Deep Learning per controllare che i valori dell'ultimo layer non divergano. Infatti usando questa funzione i Q-value sommano sempre a 1.
   - *funzione di reward*: in questo caso si è provato a moltiplicare direttamente R_p e R_d (12) per verificare se avesse potuto impattare positivamente l'andamento della funzione di reward nel tempo, portando l'agente ad adottare una policy migliore.

$$
\begin{aligned}
R_p &= \begin{cases} c, & \text{se il veicolo vince l'asta} \\ (p - P) & \text{altrimenti} \end{cases} \\
R_d &= 1 - a \\
R_{tot} &= R_p * R_d
\end{aligned} \tag{12}
$$

4.6.4 Algoritmo di training

L'algoritmo 3 descrive ad alto livello la procedura con cui viene effettivamente applicato lo sconto alla valuta che il veicolo preleva dal wallet. La primitiva `predictBid` viene descritta nel dettaglio nell'algoritmo 4.
La funzione `gatherEnvData` è la primitiva che si occupa di rilevare la situazione del traffico nelle vicinanze del veicolo: restituisce in output il vettore stato (9). Entrambe le funzioni sono in realtà metodi di una classe padre comune legata al veicolo stesso. Si considerino quindi tutte le variabili con prefisso `self.` come attributi di classe, che hanno quindi uno *scope* globale. In questo modo ad ogni passo di simulazione è possibile accedere allo stato dell'ambiente e all'azione svolta nel passo precedente (`self.prevState`, `self.prevAction`). Anche lo stesso `bidder` è un attributo di classe, diventando così facilmente accessibile da diverse parti del codice.
In algoritmo 4 nelle righe da 1 a 12 la funzione si occupa di aggiornare la memoria del bidder ed eventualmente di allenare il modello, mentre da riga 13 a 15 viene inserito lo stato attuale nella rete per richiedere lo sconto da applicare.
Il metodo `remember` aggiorna l'experience replay, aggiungendo la tupla specificata in

input. Il metodo `updateTargetModel` invece copia i pesi del q-network aggiornando i pesi del target-network.

Algoritmo 3 discountBid

Input

w: valuta prelevata per la puntata o sponsorship

Output

f: puntata o sponsorship scontata

Variabili

bidder: self.bidder, agente che applica gli sconti

1: $s \leftarrow$ gatherEnvData()
2: $a \leftarrow bidder$.predictBid(s)
3: $discount \leftarrow a/10$
4: $f \leftarrow w * discount$

4.6.5 Primitive

In Figura 10 viene riportato il codice del metodo `train` della classe `Agent`. Quest'ultima contiene le definizioni degli iperparametri del modello e tutti i metodi utili per l'inizializzazione e l'allenamento del bidder. La funzione `train` viene chiamata dal modulo principale una volta ogni F azioni eseguite dall'agente. F è un iperparametro che ai fini degli esperimenti è stato impostato a 10. Allenando il modello ogni F azioni salvate in memoria si aggiunge variabilità nel set di dati di training a ogni iterazione. Ovviamente l'allenamento inizia una volta che l'agente ha accumulato abbastanza esperienza, cioè quando il numero di tuple salvate in memoria supera il valore della batch size.

In Figura 11 è riportato il codice del metodo `act` della classe `Agent`. É la funzione che si occupa di decidere tra exploitation e exploration, restituendo l'indice dell'azione che massimizza il q-value o di un'azione casuale altrimenti. Nel modulo principale il valore restituito dalla funzione viene poi diviso per 10 in modo da ottenere direttamente lo sconto da applicare alla puntata.

Algoritmo 4 predictBid

Input

s_{t+1}: stato all'istante t+1

Output

a: indice dell'azione restituito dal bidder

Variabili

s_t: self.prevState

a: self.prevAction

bidder: self.bidder

F: frequenza di training

K: frequenza di update del target network

```
1: r ← getReward(s_t, a, s_{t+1})
2: bidder.remember(s_t, a, r, s_{t+1})
3: self.sample ← self.sample + 1
4: if self.sample > F then
5:     self.sample ← 0
6:     self.trainCount ← self.trainCount + 1
7:     bidder.train()
8:     if self.trainCount == K then
9:         self.trainCount ← 0
10:        bidder.updateTargetModel()
11:    end if
12: end if
13: a ← bidder.act(s_{t+1})
14: self.prevState ← s_{t+1}
15: self.prevAction ← a
```

```python
def train(self):
    # estrazione di un campione dell'experience replay
    batch = random.sample(self.experience_replay, self.batch_size)

    # per ogni esempio estratto
    for state, action, reward, next_state in batch:
        # viene applicata la q-learing rule
        prediction = self.q_network.predict(state, verbose=0)
        target = self.target_network.predict(next_state, verbose=0)
        prediction[0][action] = reward + self.gamma * np.amax(target)
        # viene salvato il valore della loss per l'analisi della convergenza
        history = LossHistory()
        # allenamento della rete
        self.q_network.fit(state, prediction, epochs=1, callbacks=[history])
        with open("loss.txt", "a") as f:
            for n in history.losses:
                f.write(str(n) + "\n")

```

Figura 10: Funzione di training del bidder, codice Python.

```python
def act(self, state):
    if np.random.rand() <= self.epsilon:
        return random.randrange(self.action_size)
    q_values = self.q_network.predict(state, verbose=0)
    return np.argmax(q_values[0])


```

Figura 11: Metodo `act`, codice Python.

5 Esperimenti

Sono state svolte numerose simulazioni per verificare che entrambe le prosposte discusse in questa tesi fossero efficaci e dessero un contributo allo stato dell'arte. Nella sezione 4.5 sono riportati i risultati degli esperimenti sui veicoli di emergenza, mentre nella sezione 5.2 si discutono i risultati dell'allenamento e del successivo testing del gestore della valuta.

5.1 Simulazioni con veicoli di emergenza

Durante gli esperimenti è stato scelto un veicolo a priori che avesse un percorso che attraversasse un numero sufficiente di incroci interni. In Tabella 2 viene mostrata la configurazione usata: si noti che viene fatto variare il numero di veicoli (80-170) presenti nella mappa insieme al veicolo d'emergenza, per un totale di 9 simulazioni da 5000 iterazioni l'una.

Tabella 2: Configurazione

Variabile	Valore
Durata	5000
Numero di veicoli	80-170
Model	COMP
Enhancement	True
Sponsorship	True
β (sponsorship)	0.2
Payment policy	Only Winner Pays

In Figura 12 le due curve descrivono l'incremento del tempo di attesa totale nel traffico dell'EV al variare del numero di veicoli generati all'inizio della simulazione, con e senza il sound boost attivo. I dati relativi al tempo di attesa agli incroci non vengono riportati in quanto non significativi: l'EV passa sempre per primo vincendo sempre le aste. Come si può notare la curva blu (media del tempo di attesa nel traffico del veicolo d'emergenza) è sempre sotto la curva rossa: viene limitato soprattutto l'aumento vertiginoso dei tempi (quando il sound boost è disattivato) che si verifica

passando da 130 a 140 veicoli, cioè al limite di saturazione della mappa. In queste condizioni infatti la simulazione inizia a contenere troppi veicoli, e le condizioni del traffico peggiorano inevitabilmente.

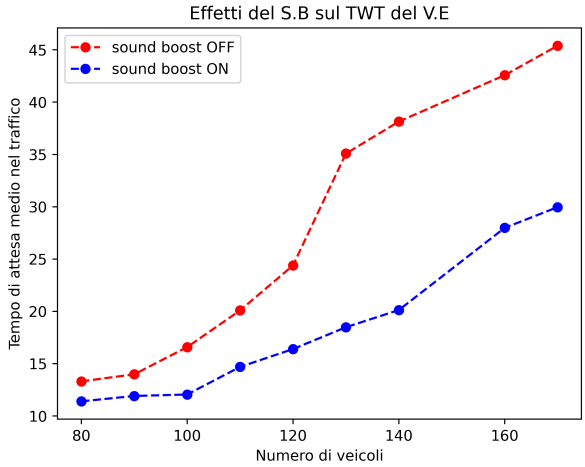

Figura 12: Media dei tempi di attesa dell' EV in base al numero di veicoli

É stato importante anche verificare che l'esperienza di guida degli altri veicoli non venisse impattata in modo invasivo, deteriorando troppo quella dei veicoli che non ricevono il boost rispetto a chi l'ha ricevuto almeno una volta sul suo percorso. Per questo si sono misurate le differenze nel TWT nel CWT per le due categorie di veicoli attivando o disattivando il sound boost. Sia V_t l'insieme che comprende tutti i veicoli presenti sulla mappa tranne l'EV. Sia invece $V \subseteq V_t$ l'insieme dei veicoli che riceve il sound boost almeno una volta durante il suo percorso. L'insieme dei veicoli che non riceve mai il sound boost è quindi $V' = V_t - V$. Un generico veicolo appartenente a V' potrebbe alternativamente *(i)* seguire un percorso in una zona non raggiunta dal segnale o *(ii)* seguire un percorso che comprende le stesse strade dell'EV (o almeno in parte) ma in verso opposto, quindi risentendo negativamente del sound boost (in maniera indiretta). Si consideri che i veicoli appartenenti a V' perchè non raggiunti dal segnale sono pochi, dato che devono avere un percorso che non si interseca in nessun punto con quello dell'EV (scelto in modo che abbia un percorso abbastanza ampio, che spazi su gran parte della mappa). In Figura 13 e in Figura 14 vengono mostrati i risultati degli esperimenti. Per quanto riguarda il CWT, le differenze tra i tempi in entrambe le categorie di veicoli è minima. Questo suggerisce che il tempo di

attesa medio che un veicolo appartenente a V' prevede di aspettare per il suo turno di attraversamento non aumenta di molto quando un veicolo d'emergenza che emette il sound boost è presente nella mapppa. Soltanto in situazioni di traffico intenso l'aumento può essere significativo. Similmente, il sound boost in media migliora solo leggermente i tempi dei veicoli appartenenti a V. Lo stesso ragionamento è valido per il tempo di attesa nel traffico, che però mostra una deviazione standard molto alta in situazioni di congestione. Ciò significa che i veicoli che hanno un percorso in conflitto con l'EV (ad esempio uguale ma con verso opposto) potrebbero passare molto più tempo in coda rispetto a chi si trova lontano dalle zone interessate dal segnale. Infatti i primi subiscono negativamente gli effetti del boost, mentre i secondi non ne risentono. Tuttavia questi problemi sono rilevanti soltanto superata la soglia dei 140 veicoli (soglia di saturazione).

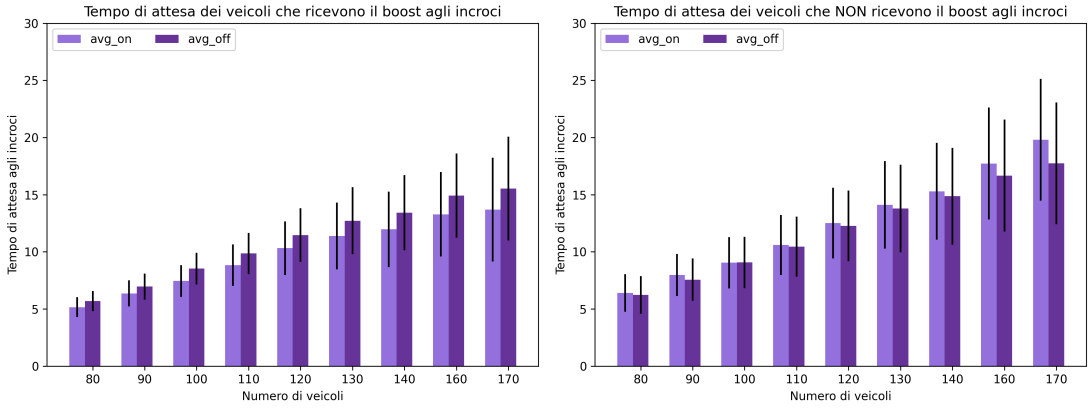

Figura 13: Media del tempo di attesa dei veicoli agli incroci. Le barre chiare indicano la media del tempo di attesa con il sound boost attivo, viceversa per le barre scure. I baffi neri indicano la deviazione standard.

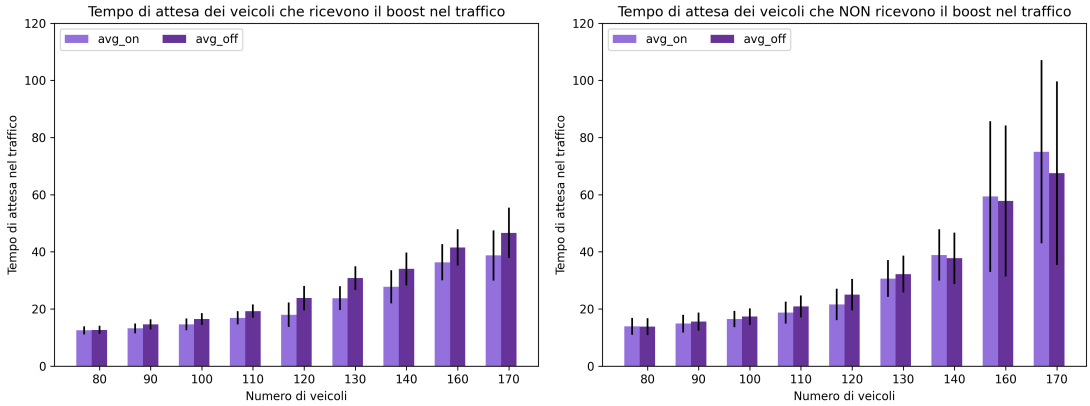

Figura 14: Media del tempo di attesa dei veicoli nel traffico. Le barre chiare indicano la media del tempo di attesa con il sound boost attivo, viceversa per le barre scure. I baffi neri indicano la deviazione standard.

5.2 Allenamento e testing del gestore della valuta

Gli esperimenti sul bidder si suddividono in una fase di allenamento (descritta nella sezione 5.2.1) e una fase di test (descritta nella sezione 5.2.2). Durante la prima i parametri interni della rete neurale vengono modificati gradualmente per cercare di massimizzare la funzione di reward nel tempo, cercando di raggiungere la politica ottima di gestione del budget. Alla fine di ogni allenamento il modello viene salvato per poi essere valutato nella fase di test, in cui invece i parametri della rete rimangono invariati.

5.2.1 Fase di allenamento

È stato necessario allenare il bidder per ogni modello e per ogni combinazione degli iperparametri. Nonostante siano stati fatti numerosi test, vengono riportati soltanto i dati relativi alle configurazioni per cui sono state misurate performance rilevanti. Ogni fase di training è durata 200000 passi di simulazione. In Tabella 3 viene riportata la configurazione usata durante l'allenamento di entrambi i modelli. In Tabella 4 vengono riportate le specifiche hardware e software della macchina su cui sono stati svolti gli esperimenti.

Tabella 3: Configurazione per il training su bidderV1 e bidderV2

Variabile	Valore
Durata	200000
Numero di veicoli	120
Model	COMP
Enhancement	True
Sponsorship	True
β (sponsorship)	0.1
c (crossing reward)	2
Payment policy	Only Winner Pays

Il tempo necessario per l'allenamento del modello non ha mai superato l'ora e trenta minuti. In Figura 15 è descritto l'andamento della funzione di reward nel tempo. Per visualizzare il grafico in forma più compatta è stata calcolata la media dell'output della funzione che restituisce la reward ogni 180 chiamate. Come ci si poteva aspettare le funzioni risultanti sono irregolari, con molti picchi e valli. Tuttavia si può osservare in entrambe le funzioni una ripida salita nelle fasi iniziali, per poi passare ad una fase molto variabile, arrivando infine a una nuova fase di risalita. E' piuttosto comune osservare questo tipo di comportamento nelle funzioni di reward di modelli in cui la reale valutazione del comportamento dell'agente nella realtà si baserebbe su sequenze di azioni contigue nel tempo [14]. Maggiori dettagli verrano forniti nella sezione 6.2.

Tabella 4: Caratteristiche Hardware e Software della macchina sperimentale.

Variabile	Valore
CPU	Intel(R) Core(TM) i5-8400 CPU @ 2.80GHz
GPU	NVIDIA GeForce GTX 1060 6GB
RAM	8Gb
OS	Ubuntu 22.04.3 LTS $x86_{64}$

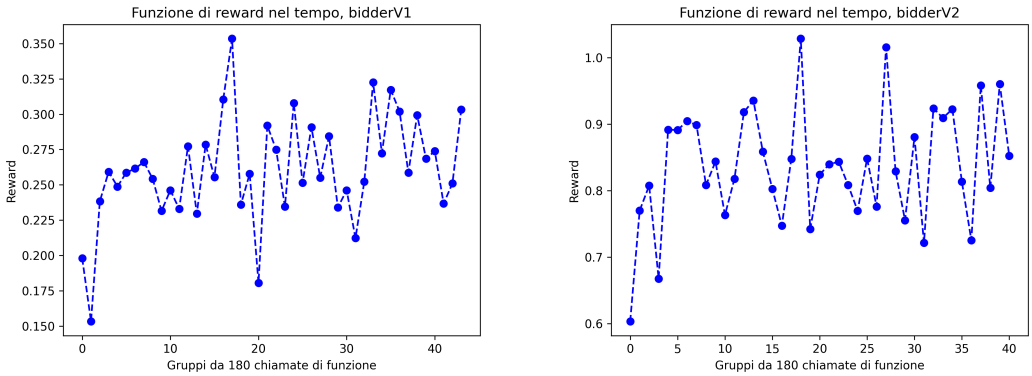

Figura 15: Andamento della funzione di reward durante la fase di allenamento di bidderV1 e bidderV2 rispettivamente. L'asse X rappresenta il tempo, misurato in chiamate alla funzione che restituisce la reward. Sull'asse delle Y viene riportata la media ogni 180 chiamate di funzione.

5.2.2 Fase di test

È stato svolto un primo *set* di esperimenti su entrambi 'bidderV1' e 'bidderV2', per verificare che l'ammontare di valuta risparmiata fosse significativo. Per farlo si è osservato il budget con cui il veicolo fa ritorno al punto di partenza alla fine di ogni corsa (secondo il metodo fair classico questa quantità è sempre pari a zero). In Tabella 5 viene riportata la configurazione usata. In particolare è stato fatto variare il numero di veicoli, e per ogni valore fissato sono state svolte 10 simulazioni. Per entrambi i modelli si osserva in media un'alta percentuale di valuta risparmiata, che diminuisce all'aumentare del numero di veicoli nella simulazione, come prevedibile. I tempi di attesa agli incroci risultano leggermente più alti quando il bidder è attivo, favorendo invece i tempi di attesa nel traffico. Per entrambi i modelli i tempi di attesa totali rimangono essenzialmente invariati rispetto al metodo fair. Tra i due modelli bidderV1 si rivela essere più efficiente: apparentemente usare una media pesata nella funzione di reward dando più importanza a R_d permette all'agente di imparare una strategia migliore.

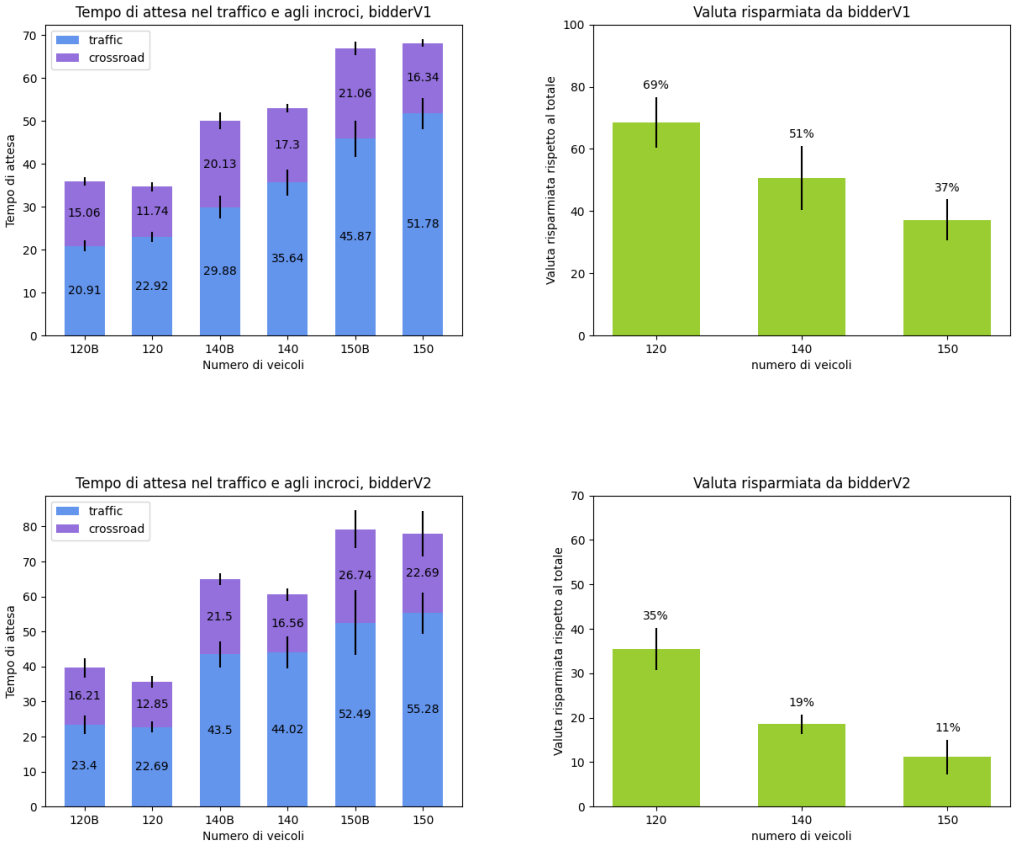

Figura 16: Nella parte alta sono riassunti i dati relativi a bidderV1, nella parte bassa i dati sono relativi a bidderV2. A sinistra, tempi di attesa del veicolo con e senza gestione della valuta (il pedice 'B' indica che il bidder è attivato nelle simulazioni). Le porzioni di barra scure indicano il tempo di attesa agli incroci, mentre le porzioni chiare rappresentano il tempo di attesa nel traffico. A destra viene mostrata la percentuale della valuta risparmiata durante la corsa al variare del numero di veicoli. In entrambi i grafici, le barre nere indicano la deviazione standard.

Tabella 5: Configurazione per i test su bidderV1

Variabile	Valore
Durata	5000
Numero di veicoli	120,140,150
Model	COMP
Enhancement	True
Sponsorship	True
β (sponsorship)	0.1
Payment policy	Only Winner Pays

Un secondo set di esperimenti è stato svolto confrontando bidderV1 (che apparentemente performa meglio di bidderV2) con un *Random bidder* che sceglie uno sconto casuale da applicare su ogni puntata e sponsorship. Si è pensato che una *baseline* meno deterministica rispetto al metodo *fair* potesse essere più significativa nella fase di valutazione del modello. Si è anche voluto verificare che la policy imparata fosse resistente ad un cambio di strategia di bidding degli altri veicoli nella simulazione. Infatti oltre a far variare il numero di veicoli si è svolta un'ulteriore simulazione in cui $\beta = 0.2$, in modo da raddoppiare l'ammontare di valuta massima che le altre auto possono puntare durante una sponsorship. Si è pensato che aumentare ulteriormente il valore di β non sarebbe stato significativo, dato che i veicoli avrebbero potuto allocare troppa della loro valuta a disposizione per una singola sponsorship. Le performance di bidderV1 sono compatibili con quelle misurate nel primo set di esperimenti, e in tutti i casi sono nettamente migliori rispetto a quelle del random bidder. L'ammontare di valuta risparmiata inizia a diminuire soltanto nel caso $\beta = 2$ quando vengono simulati 140 veicoli.

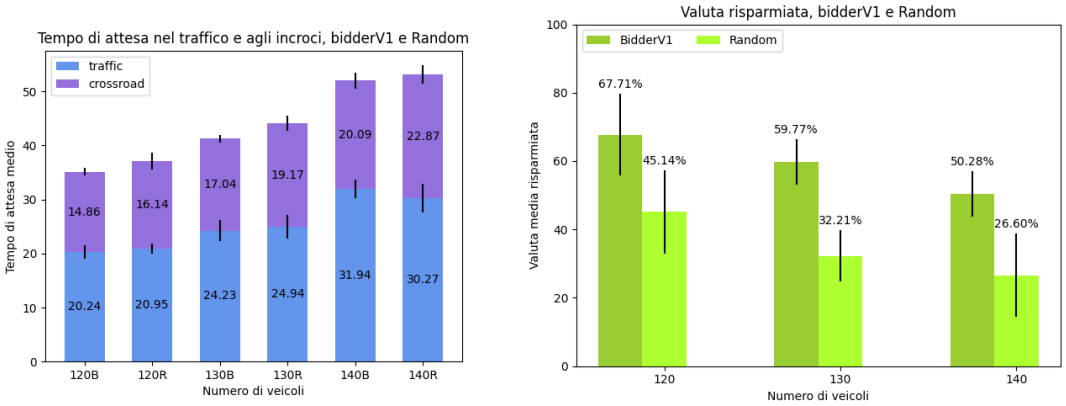

Figura 17: Tempi di attesa (a sinistra) e valuta risparmiata (a destra) con $\beta = 0.1$. Le barre verdi scure indicano la percentuale di valuta risparmiata da bidderV1, mentre le verdi chiare si riferiscono ai risparmi del random bidder.

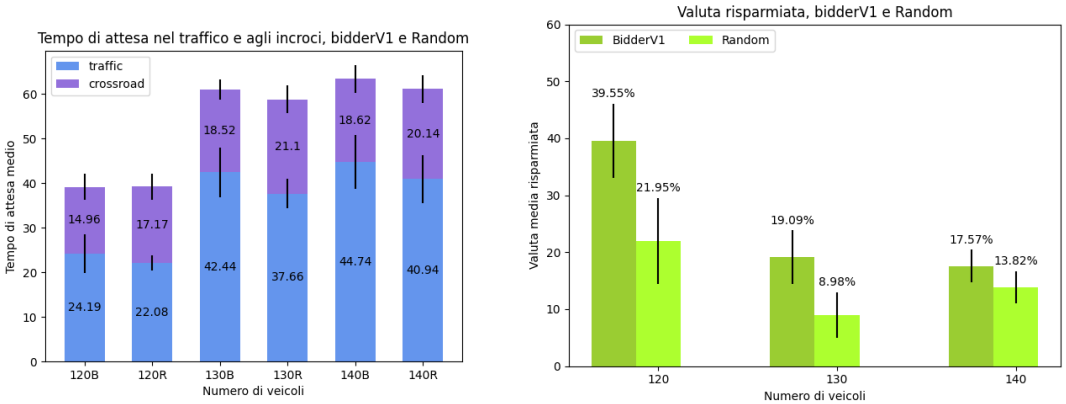

Figura 18: Tempi di attesa(a sinistra) e valuta risparmiata (a destra) con $\beta = 0.2$.

6 Conclusioni

In questa tesi si sono studiati alcuni dei problemi centrali degli algoritmi ad asta per la gestione degli incroci. Dopo aver presentato in modo approfondito lo stato dell'arte, sono stati proposti e discussi:

- un modo per migliorare i tempi di attesa dei veicoli di emergenza, i cui risultati vengono discussi nella sezione 6.1;
- una strategia innovativa per imparare strategie di gestione della valuta basandosi sull'esperienza dei veicoli nel traffico. Le conclusioni riguardo questa parte vengono tratte nella sezione 6.2.

6.1 Conclusioni sulla gestione dei veicoli di emergenza

Il miglioramento dei tempi di attesa dell'EV anche in situazioni di traffico intenso dato dal *Sound Boost* è sicuramente promettente. In un futuro in cui tutti i veicoli saranno connessi tra di loro e con l'infrastruttura circostante, si pensa che la propagazione di segnali di potenziamento come quello proposto sia possibile. I risultati sono positivi anche dal punto di vista degli altri veicoli: i tempi di attesa peggiorano soltanto moderatamente per i veicoli che non ricevono il *Sound Boost*, e migliorano leggermente per chi lo riceve almeno una volta durante il suo percorso. Ciò significa che il sound boost agisce in modo non invasivo sul traffico, evitando di creare situazioni spiacevoli di congestione.

6.2 Conclusioni sul gestore della valuta

Come si può notare l'uso del bidder per la gestione della valuta a disposizione non impatta in modo significativo i tempi di attesa, come desiderabile. Tendenzialmente si osserva un leggero aumento del tempo di attesa agli incroci in favore del tempo di attesa nel traffico. Questa variazione probabilmente è dovuta alla diversa gestione delle sponsorship quando il bidder è attivo rispetto a quando non lo è (7, 3).

Infatti partire dallo stesso valore su cui applicare lo sconto sia per le puntate che per le sponsorship rende il veicolo meno efficiente nell'attraversare gli incroci per primo, dato che spende molta più valuta durante le sponsorizzazioni nel traffico rispetto al normale (in cui sceglie casualmente una piccola percentuale del suo budget). Un altro motivo potrebbe essere legato alla funzione di reward: per l'agente allocare le sue risorse sulle sponsorship rispetto alle puntate è apparentemente più efficace che fare l'inverso.

È incoraggiante la stabilità del modello all'aumentare del numero di veicoli. Come prevedibile l'ammontare di valuta risparmiata diminuisce, senza però scendere sotto quella del random bidder. Anche la deviazione standard è accettabile, rendendo la predizione sull'ammontare risparmiato su una particolare corsa relativamente precisa.

In entrambi i casi le performance di bidderV1 sono molto migliori rispetto a quelle del random bidder. In particolare si nota come per $\beta = 0.1$ il modello sia particolarmente resistente anche in condizioni di traffico intenso (140 veicoli). Per $\beta = 0.2$ invece il bidder inizia a diminuire la sua efficienza nelle stesse condizioni. Questo problema potrebbe essere affrontato in un lavoro futuro: con abbastanza risorse hardware si potrebbe lasciare il modello in allenamento facendo variare dinamicamente la politica di gestione del budget degli altri veicoli, e al contempo misurare la performance. In questo modo si potrebbe verificare l'effettiva adattabilità del modello nel caso in cui gli altri veicoli cambino il loro comportamento.

La forte non linearità e non convessità del modello rendono la reward molto instabile, ma nonostante questo la misura di performance scelta è comunque a buoni livelli. Per migliorare la stabilità della funzione di reward si potrebbe assegnare la ricompensa all'agente sulla base di una sequenza di stati e azioni contigue nel tempo, come $< s_1, a_1, s_2, a_2, ..., s_{t-1}, a_t >$ come spiegato in [14]. In questo modo l'agente manterrebbe in memoria delle esperienze nel traffico su periodi più lunghi (quindi teoricamente più significativi) invece che limitarsi alla singola puntata o sponsorship.

Capire qual'è la causa delle decisioni prese da modelli di questo tipo è un problema intrinseco delle reti neurali, sistemi *black-box* per eccellenza. In certi casi però osservando attentamente l'output si può comunque provare a trarre delle conclusioni

significative. Il grafico in figura 19 è riferito ad una serie di misurazioni che sono state svolte in un caso particolare, cioè in cui il bidder è riuscito a risparmiare con successo circa il 70% della sua valuta. In particolare si è calcolato il numero totale di auto che sono state incontrate dal veicolo di testing sul suo percorso e l'ammontare di valuta spesa per ogni incrocio in puntate e sponsorship. Come si può notare, il numero di veicoli incontrati e la valuta spesa sono in qualche modo inversamente correlati. L'incrocio in cui vengono incontrate meno auto (H) è in realtà quello in cui il bidder alloca più valuta. Un comportamento simile era in realtà prevedibile, anche se non scontato: puntare più valuta dove ci sono meno agenti in competizione massimizza il peso che la puntata stessa ha sulla media delle puntate, aumentando le probabilità del veicolo di aggiudicarsi l'attraversamento o una posizione migliore nel traffico.

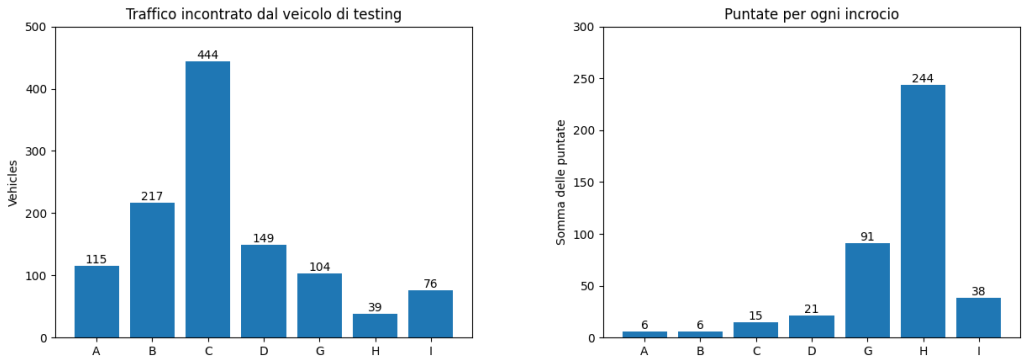

Figura 19: A sinistra, il numero di veicoli incontrati dal veicolo di testing per ogni incrocio. A destra, l'ammontare di valuta allocata dal bidder per ogni incrocio.

6.3 Sviluppi Futuri

Un problema del sistema auction based classico discusso in questa tesi riguarda il *throughput* degli incroci: facendo attraversare soltanto un auto per volta non si ottimizzano le risorse spaziali offerte dall'infrastruttura stradale. Tuttavia alcune ricerche recenti [15] hanno dimostrato che gli algoritmi basati sulle aste possono migliorare nettamente i tempi di percorrenza e la fairness se utilizzati per la gestione dei semafori, aumentando il numero di auto che attraversano l'incrocio per unità di

tempo. In questa variante tutti i veicoli che attendono in coda effettuano una puntata per ottenere il verde, quindi l'attraversamento dell'incrocio. In tal modo si potrebbero reimpiegare i dispositivi già presenti sulle nostre strade, equipaggiandoli con sistemi intelligenti in grado di interagire tra di loro e con i veicoli stessi. I metodi di gestisone del budget e di ottimizzazione degli EV proposti potrebbero quindi rivelarsi adatti ed efficaci anche in questi contesti.

7 Riferimenti bibliografici

[1] Cabri, G., Gherardini, L., Montangero, M. Auction-based crossings management. In Proceedings of the 5th EAI International Conference on Smart Objects and Technologies for Social Good (pp. 183-188),2019.

[2] Cabri, G., Gherardini, L., Montangero, M. et al. About auction strategies for intersection management when human-driven and autonomous vehicles coexist. Multimed Tools Appl 80, 15921–15936, 2021.

[3] Carlino, D., Boyles, S. D., Stone, P. Auction-based autonomous intersection management. In 16th International IEEE Conference on Intelligent Transportation Systems (pp. 529-534). IEEE, 2013.

[4] Dresner, K., Stone, P. Multiagent traffic management: An improved intersection control mechanism. In Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems (pp. 471-477), 2005.

[5] Glorio, N., Mariani, S., Cabri, G., Zambonelli, F. An Adaptive Approach for the Coordination of Autonomous Vehicles at Intersections. In 2021 IEEE 30th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE) (pp. 1-6). IEEE, 2021.

[6] Gherardini, L., Cabri, G., Montangero, M. Decentralized approaches for autonomous vehicles coordination. Internet Technology Letters, e398, 2022.

[7] `https://www.istat.it/it/archivio/286933`, ultimo accesso: 2023-09-05

[8] `https://www.sae.org/standards/content/j3016_202104/`, ultimo accesso: 2023-09-30

[9] `https://eclipse.dev/sumo/`, ultimo accesso: 2023-09-05.

[10] `https://sumo.dlr.de/docs/TraCI.html`, ultimo accesso: 2023-09-05.

[11] https://www.openstreetmap.org/#map=6/42.090/12.656, ultimo accesso: 2023-09-05.

[12] https://keras.io/, ultimo accesso: 2023-09-17.

[13] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324, 1998.

[14] Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." arXiv preprint arXiv:1312.5602, 2013.

[15] Muzzini, F., Capodieci, N., Montangero, M. Coordinated traffic lights and auction intersection management in a mixed scenario. In Proceedings of the 2023 ACM Conference on Information Technology for Social Good (pp. 1-1), 2023.

[16] Joo, H., Ahmed, S. H., Lim, Y. Traffic signal control for smart cities using reinforcement learning. Computer Communications, 154, 324-330, 2020.

[17] Kaelbling, L. P., Littman, M. L., Moore, A. W. Reinforcement learning: A survey. Journal of artificial intelligence research, 4, 237-285, 1996.

[18] Karthikeyan, P., Chen, W. L., Hsiung, P. A. Autonomous Intersection Management by Using Reinforcement Learning. Algorithms, 15(9), 326, 2022.

[19] Krizhevsky, A., Sutskever, I., Hinton, G. E. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25, 2012.

[20] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... Hassabis, D. Mastering the game of Go with deep neural networks and tree search. nature, 529(7587), 484-489, 2016.

8 Ringraziamenti

Ringrazio innanzitutto i prof. Cabri e Montangero, per avermi seguito durante il lavoro ed essersi interessati al progetto. Un grazie particolare a Luca Gherardini, per aver reso disponibile il suo software pubblicamente e avermi aiutato nella parte tecnica. Grazie anche alla prof. Giorgia Franchini per avermi dato consiglio sulla parte relativa al Deep Learning.

Grazie mamma e papà, perchè mi avete dato l'opportunità di studiare. Grazie perchè mi siete stati sempre vicino indipendentemente dal percorso che ho voluto scegliere, sia universitario che di vita.
Grazie Marti, che mi hai accompagnato dall'inzio alla fine. Senza di te non sarei qui: non conosco persona più forte di te, mi sproni ogni giorno a essere la versione migliore di me stesso.
Grazie Dotti e Erry, per fortuna ongi tanto ci siete voi a farmi staccare la spina con una partita a magic o a D&D.
Grazie Carlo, le nostre risate risuoneranno al FIM ancora per un po'. Se sono arrivato fin qua è anche grazie ai nostri progetti di gruppo, che bene o male son sempre piaciuti.
Grazie Fede, sei stato un grande mentore, sono sicuro che diventerai un ottimo professore.
Grazie agli amici dell'associazione ludica "La Contea" e del "Decumano Ovest", con cui si sono passati bei momenti di condivisione e di spensieratezza durante questi anni.

Questo giorno non appartiene a un uomo solo, ma a tutti. Insieme ricostruiamo questo mondo, da poter condividere nei giorni di pace.

ARAGORN