Università degli Studi di Salerno

Corso di Intelligenza Artificiale

Ottimizzazione del Flusso del Traffico Urbano mediante l'applicazione di Q-learning

Progetto di Ferriero Raffaele e Nisivoccia Giuseppe

1.Introduzione

Il progetto si è posto l'obiettivo di sviluppare e implementare un sistema per il controllo del traffico urbano, inizialmente concepito come un approccio basato su Deep Q-learning. Tuttavia, durante la fase di implementazione, sono emerse sfide legate alla compatibilità della libreria selezionata.
Di conseguenza, l'approccio è stato riorientato esclusivamente verso l'utilizzo dell'algoritmo Q-learning.
Questo sistema mira a ottimizzare:

- Il flusso veicolare
- Ridurre la congestione stradale
- Massimizzare l'efficienza complessiva della circolazione urbana.

L'obiettivo è sviluppare un sistema intelligente in grado di adattare dinamicamente i tempi dei semafori in base al traffico attuale. La priorità è ridurre al minimo i tempi di attesa dei veicoli e migliorare il funzionamento generale del sistema.
Il progetto si basa sull'analisi accurata delle condizioni presenti nelle intersezioni stradali, includendo la densità del traffico e la presenza di veicoli in attesa. Inizialmente, era stata considerata l'opzione di integrare le condizioni meteorologiche, ma l'ambiente di lavoro non lo ha reso possibile.
In aggiunta, è stato utilizzato un sistema di ricompense e penalità cruciale per guidare l'apprendimento del sistema. Questo sistema premia le decisioni che migliorano l'efficienza nel controllo del traffico e sanziona le azioni che causano rallentamenti o aumentano i tempi di attesa.
Per la configurazione dell'ambiente, la gestione delle osservazioni, delle azioni e delle ricompense, è stata adottata la libreria sumo-rl.

2.Stato dell’arte

Prima di passare all’effettiva implementazione sono stati visionati vari paper presenti in letteratura e tra i vari sfogliati si riportano:

- **JADE, TraSMAPI and SUMO: A tool-chain for simulating traffic light control**: L'articolo affronta le problematiche legate alla congestione del traffico, proponendo una catena di strumenti open-source per sviluppare soluzioni basate su multiagenti per gestire il traffico urbano. Utilizzando JADE per sistemi multiagente e SUMO per simulare le interazioni del traffico, l'articolo dimostra come il framework consenta di sperimentare diverse soluzioni intelligenti di trasporto. Attraverso l'impiego del Q-Learning in una rete simulata di semafori, si evidenzia l'efficacia dell'approccio proposto.

- **Using a Deep Reinforcement Learning Agent for Traffic Signal Control:** Il testo presenta un nuovo sistema di controllo dei segnali stradali che sfrutta grandi quantità di dati di alta qualità per migliorare l'efficienza del trasporto. Utilizzando tecniche avanzate di deep learning, crea un agente di controllo dei segnali stradali altamente adattivo in SUMO, un simulatore di traffico. Introduce un nuovo metodo di rappresentazione dello stato del traffico, addestra una rete neurale convoluzionale profonda con Q-learning e dimostra che l'agente proposto riduce significativamente il ritardo, la lunghezza delle code e il tempo di viaggio rispetto ad altre soluzioni basate su reti neurali più semplici.

- **A Reinforcement Learning Approach for Intelligent Traffic Signal Control at Urban Intersections:** Il testo propone un metodo basato sul reinforcement learning per il controllo dei semafori nelle intersezioni urbane. Utilizza reti neurali per gestire le complesse dinamiche del traffico, considerando spazi di stati e di azioni ampi e discreti. Sfrutta dati real-time sul traffico e varie configurazioni di simulazione per addestrare un modello adattabile a diverse condizioni. I risultati delle simulazioni mostrano che questo approccio supera significativamente altri metodi di controllo dei semafori, riducendo lunghezze delle code e tempi di attesa.

- **Real-time deep reinforcement learning based vehicle navigation:** Il testo propone l'utilizzo del deep reinforcement learning per creare un sistema di navigazione intelligente in tempo reale per veicoli, affrontando la congestione del traffico. Si formulano decisioni sequenziali per la navigazione dei veicoli, integrando agenti intelligenti in SUMO per simulare nove scenari di traffico realistici. I risultati dimostrano la convergenza efficiente degli agenti di navigazione dei veicoli e la loro capacità di prendere decisioni ottimali in condizioni di traffico mutevoli. Il metodo proposto supera gli algoritmi di ottimizzazione dell'instradamento di riferimento, evidenziando maggiori miglioramenti in scenari con strade più numerose e traffico più complesso.

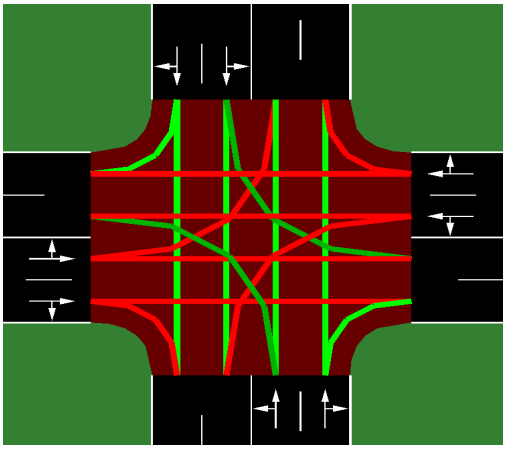
3.Definizione degli stati e delle azioni

3.1 Stati dell’ambiente

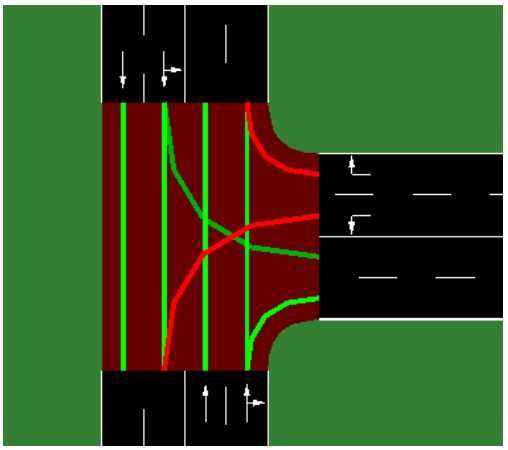
Nelle simulazioni condotte, la velocità dei veicoli è stata mantenuta ai valori predefiniti, mentre le condizioni meteorologiche, non presenti in sumo, sono state omesse per la natura specifica dell'ambiente di simulazione. Un particolare interessante riguarda la quantità approssimativa di spazio disponibile in ogni corsia, valutata attorno a otto veicoli, elemento cruciale per comprendere la fluidità e l'affollamento delle strade simulate. Inoltre, è stato configurato il tasso di generazione dei veicoli, stabilito a un veicolo per secondo, determinando così il flusso di traffico nella simulazione.

3.2 Azioni

Nel contesto del controllo del traffico, l'insieme delle azioni disponibili è discretizzato. Ogni agente responsabile del segnale stradale ha la possibilità, ogni 'delta_time' secondi, di determinare la configurazione della prossima fase verde.



Ad esempio: Nel seguente incrocio ci sono |A| = 4 azioni discrete, corrispondenti alle configurazioni di fase verde



Invece in questo incrocio ci sono |A| = 3 azioni discrete, corrispondenti alle configurazioni di fase verde.

4.Osservazioni dell’ambiente

L'osservazione predefinita per ciascun agente del segnale stradale è rappresentata da un vettore:

```
obs = [phase_one_hot, min_green, lane_1_density,...,lane_n_density, lane_1_queue,...,lane_n_queue]
```

- **phase_one_hot:** Una rappresentazione codificata in "one-hot" della fase verde attiva al momento.
- **min_green:** Una variabile binaria che indica se sono già trascorsi i secondi minimi nella fase corrente.
- **lane_i_density:** Il numero di veicoli nella corsia entrante i diviso per la capacità totale della corsia i.
- **lane_i_queue:** Il numero di veicoli in coda (con velocità inferiore a 0,1 m/s) nella corsia entrante i diviso per la capacità totale della corsia i.

5.Sistema di ricompense e penalità

La funzione di ricompensa predefinita è la variazione del ritardo cumulativo dei veicoli:

$$r_t = D_{a_t} - D_{a_{t+1}}$$

In altre parole, la ricompensa rappresenta quanto è cambiato il ritardo totale (somma dei tempi d'attesa di tutti i veicoli in avvicinamento) rispetto all'episodio precedente.

6.Metodologia di implementazione

6.1 Simulazione del traffico urbano

Il progetto è sviluppato sfruttando l'ambiente di simulazione SUMO (Simulation of Urban MObility). SUMO è un simulatore open-source altamente specializzato progettato per modellare e simulare il traffico urbano su strada. Utilizzato ampiamente nell'ambito della ricerca e dello sviluppo, SUMO offre una piattaforma flessibile e potente per eseguire simulazioni realistiche e analizzare varie strategie di gestione del traffico e dei trasporti in un contesto urbano. Essendo open-source, SUMO consente agli sviluppatori di adattare, estendere e integrare le proprie funzionalità per adempiere a una vasta gamma di esigenze di simulazione e ottimizzazione del traffico.

6.2 Sviluppo dell’agente

La classe QLAgent implementa un agente di apprendimento basato su Q-learning per l'ambiente di simulazione. Questo agente utilizza una tabella Q per memorizzare le valutazioni delle azioni in uno specifico stato e apprende iterativamente attraverso l'interazione con l'ambiente.
Variabili dell’agente:

- **state:** Lo stato attuale dell'agente.
- **state_space:** Lo spazio degli stati dell'ambiente.
- **action_space:** Lo spazio delle azioni disponibili per l'agente.
- **action:** L'azione attualmente scelta dall'agente.
- **alpha:** Il tasso di apprendimento utilizzato dall'agente.
- **gamma:** Il fattore di sconto per le ricompense future.
- **q_table:** La tabella Q che memorizza le valutazioni delle azioni in uno specifico stato.
- **exploration:** La strategia di esplorazione utilizzata dall'agente. (Epsilon-Greedy)
- **acc_reward:** La ricompensa accumulata dall'agente durante l'interazione con l'ambiente.

6.3 Training

6.3.1 Q-Learning

I parametri che utilizza l’algoritmo Q-Learning sono:

- Alpha (α): Indica il tasso di apprendimento, determina quanto un agente apprende dai nuovi dati rispetto a quelli già conosciuti.
    - Valore scelto: 0.5
- Gamma (γ): Rappresenta il fattore di sconto che controlla l'influenza delle ricompense future.
    - Valore scelto: 0.99
- Decay: Questo parametro ha come obiettivo quello di bilanciare l'esplorazione (provare nuove azioni per capire meglio l'ambiente) con lo sfruttamento (usare le azioni che sembrano essere le migliori finora)
    - Valore scelto: 1, mantenendo quindi la propensione all’esplorazione
- Runs: Rappresenta il numero di iterazioni o cicli attraverso i quali l'algoritmo Q-learning viene eseguito.
    - Valore scelto: 3
- Episodes: Indica il numero di episodi che vengono eseguiti in ciascuna iterazione.
    - Valore scelto: 10

6.3.2 SARSA

I parametri che utilizza l’algoritmo SARSA sono:

- Alpha (α): Indica il tasso di apprendimento, determina quanto un agente apprende dai nuovi dati rispetto a quelli già conosciuti.
    - Valore scelto: 0.5

- Gamma (γ): Rappresenta il fattore di sconto che controlla l'influenza delle ricompense future.
    - Valore scelto: 0.99
- Epsilon (ε): È il tasso di esplorazione. Indica la probabilità con cui l'agente sceglie un'azione casuale anziché l'azione migliore conosciuta.
    - Valore scelto: 0.2
- Lambda (λ): Regola l'importanza delle azioni passate nell'aggiornamento dei valori delle azioni.
    - Valore scelto: 1, dando quindi importanza a tutte le azioni passate
- Fourier_order: Si riferisce all'ordine delle funzioni di base di Fourier
    - Valore scelto: 7

6.4 Prestazioni

Le sessioni di test sono state condotte in tre run distinte:

- Prima Run:
    - Numero di Macchine: 100
    - Frequenza di Spawn: 1 macchina al secondo
    - Durata della Simulazione: 250 secondi

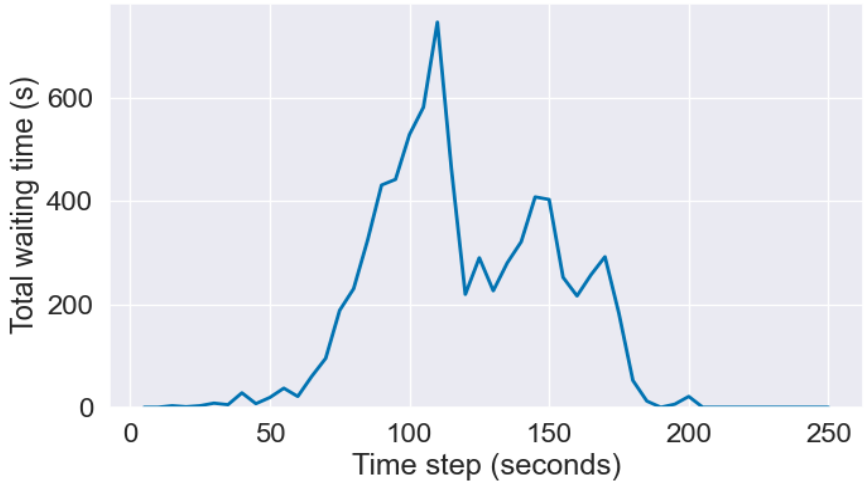
- Seconda Run:
    - Numero di Macchine: 200
    - Frequenza di Spawn: 1 macchina al secondo
    - Durata della Simulazione: 350 secondi

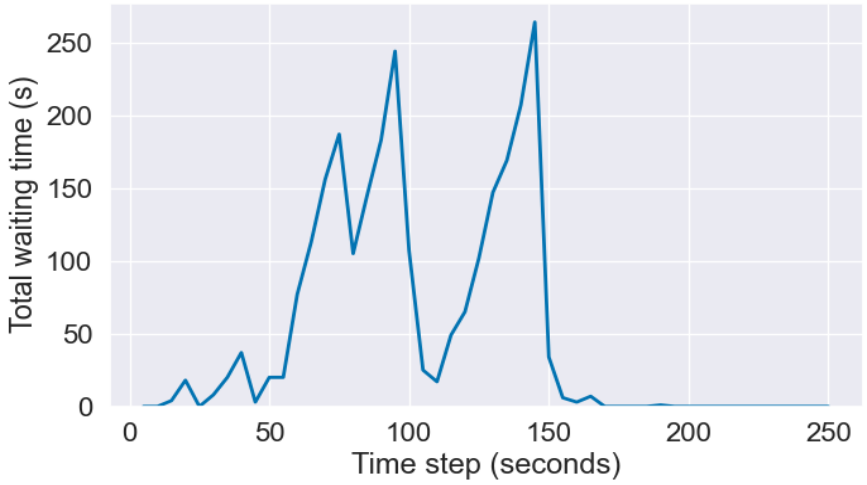
- Terza Run:
    - Numero di Macchine: 300
    - Frequenza di Spawn: 1 macchina al secondo
    - Durata della Simulazione: 450 secondi

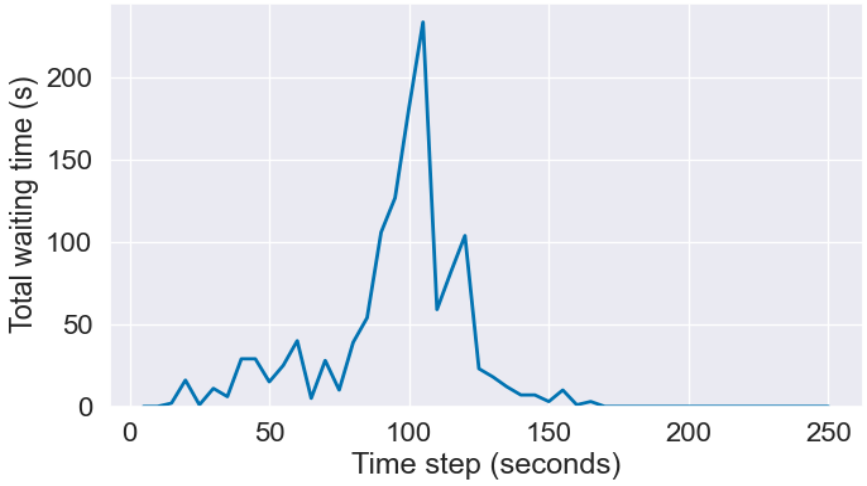
6.4.1 Q-Learning

Per comodità verranno mostrati soltanto i plot per ogni run relativi al primo, al quinto e all'ultimo episodio.

Prima Run:

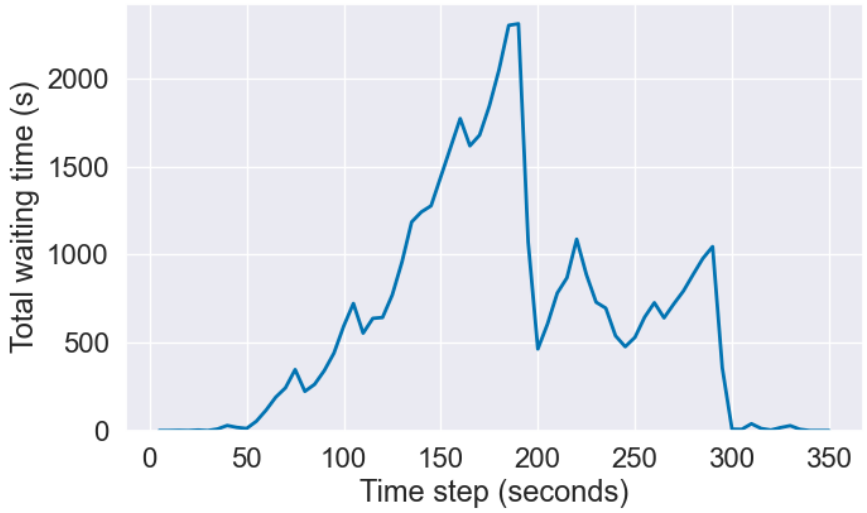


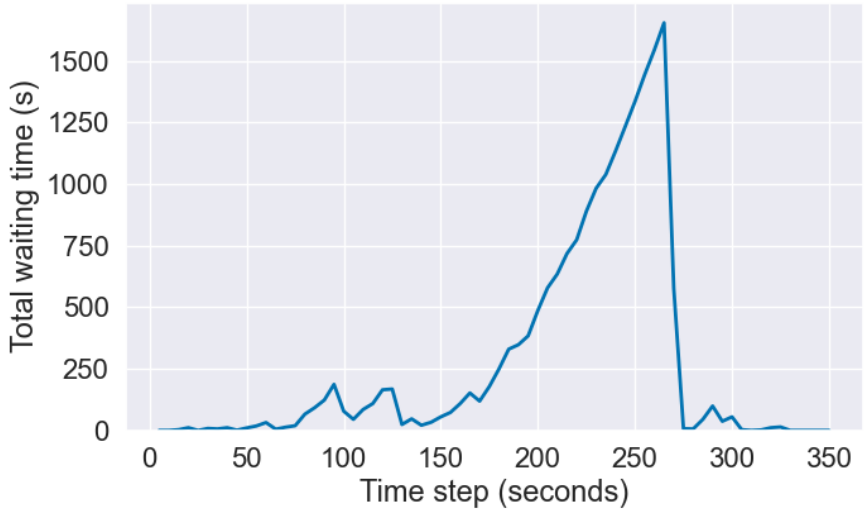


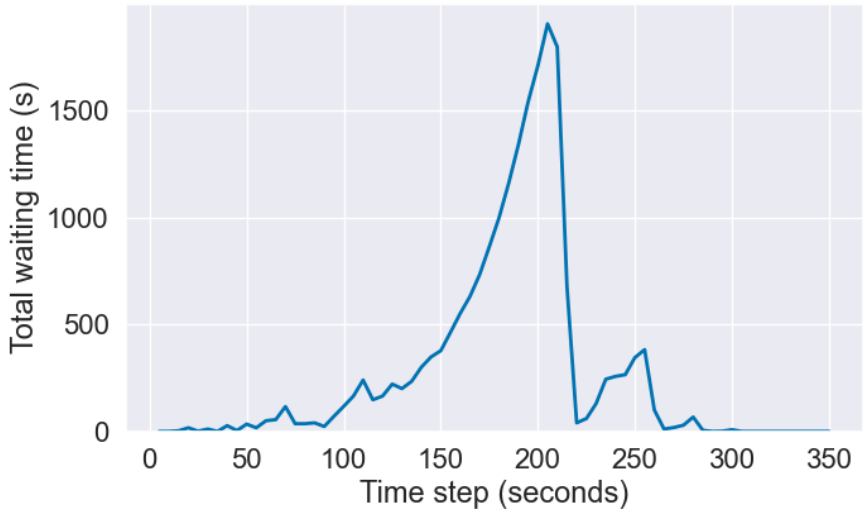


L'algoritmo Q-learning dimostra fin dalla sua prima esecuzione la capacità di ottenere risultati ottimali. Finisce gli episodi in anticipo, mantenendo brevi i tempi di attesa. Con il susseguirsi degli episodi, continua a migliorare i già eccellenti risultati.

Seconda Run:

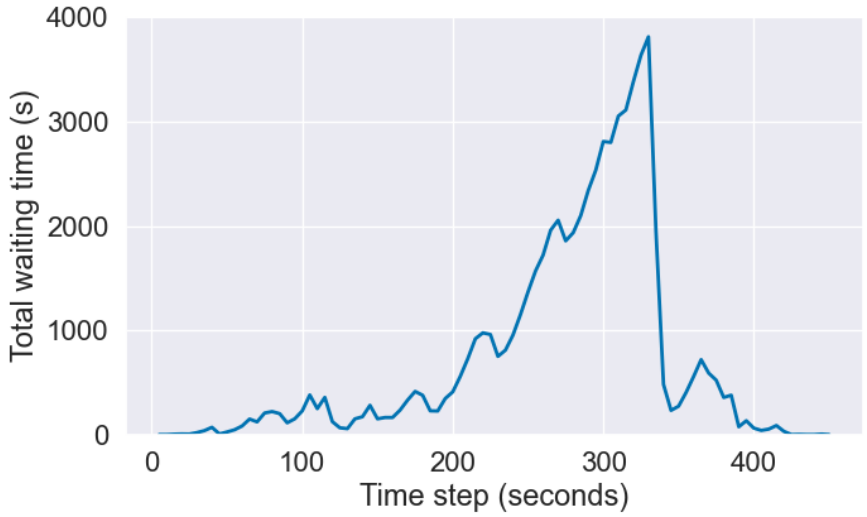


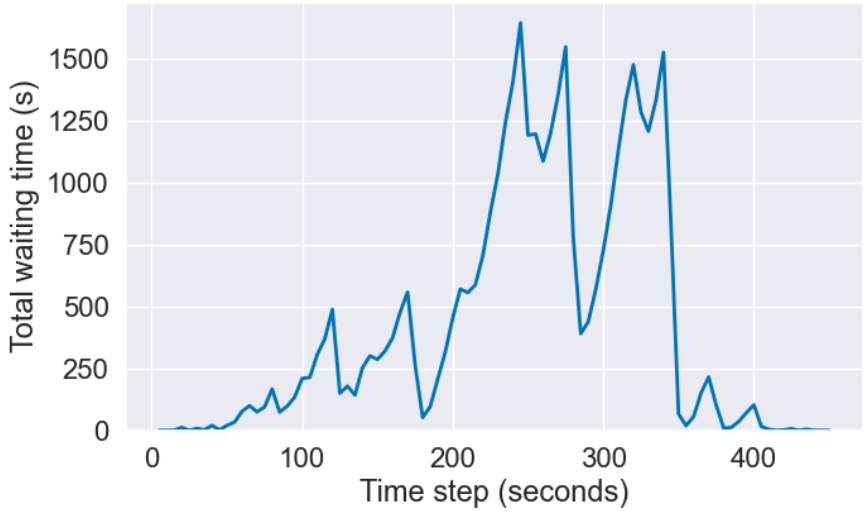


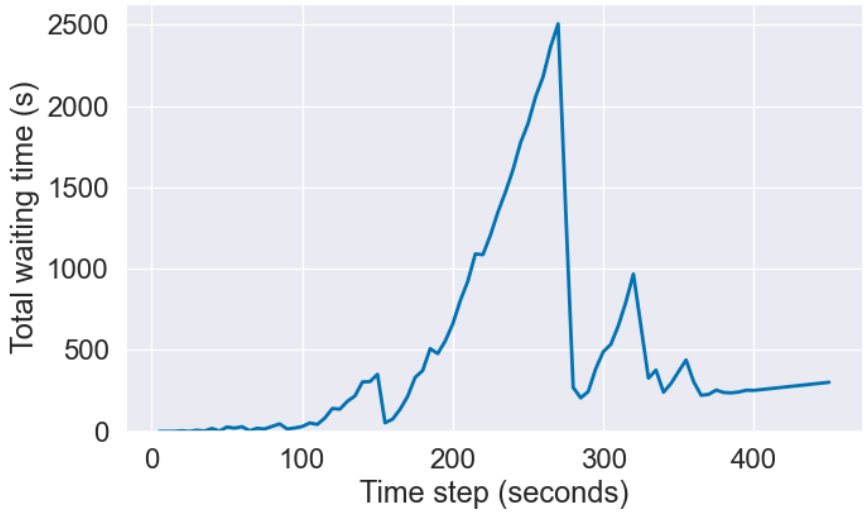


Nella seconda esecuzione, l'algoritmo Q-learning dimostra la sua capacità di gestire efficacemente il doppio dei veicoli, mantenendo bassi i tempi di attesa e concludendo gli episodi rapidamente.

Terza Run:





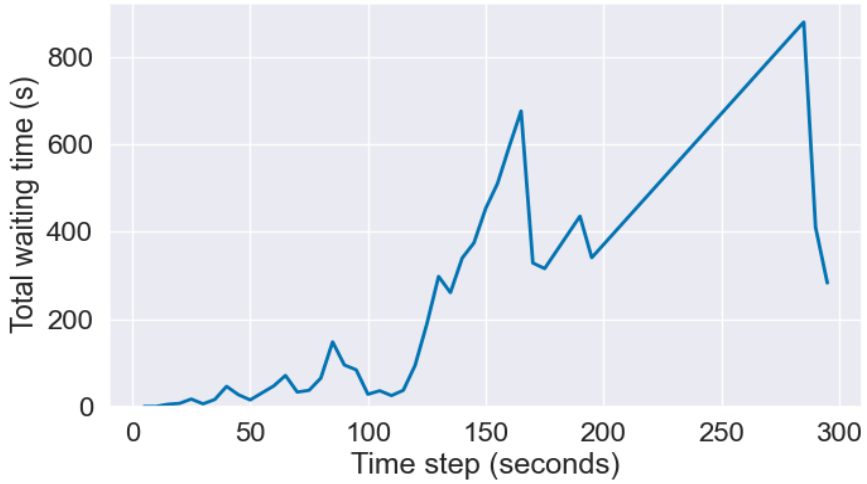


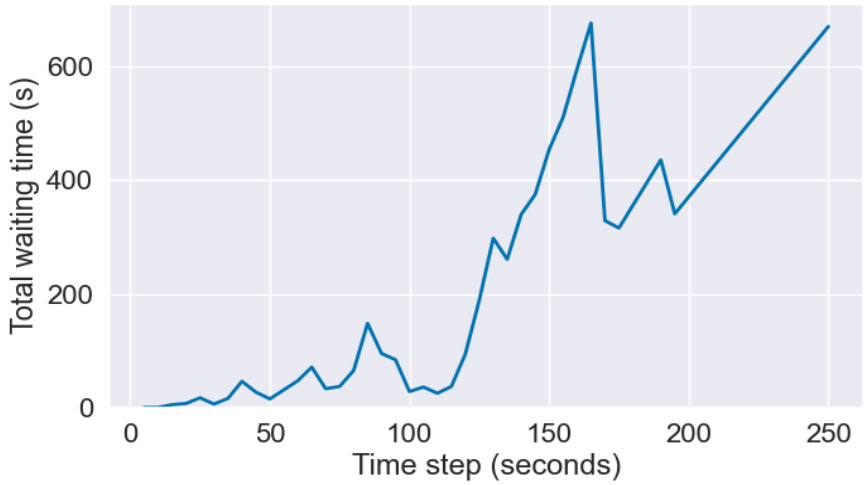
Nell'ultima esecuzione sono presenti ancora più veicoli ma i risultati non sono troppo differenti dalle run precedenti.

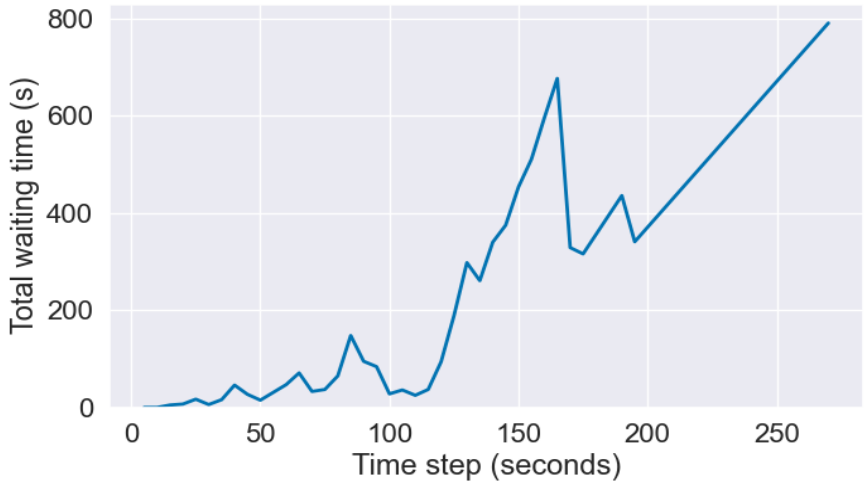
6.4.2 SARSA

Per comodità verranno mostrati soltanto i plot per ogni run relativi al primo, al quinto e all'ultimo episodio.

Prima Run:

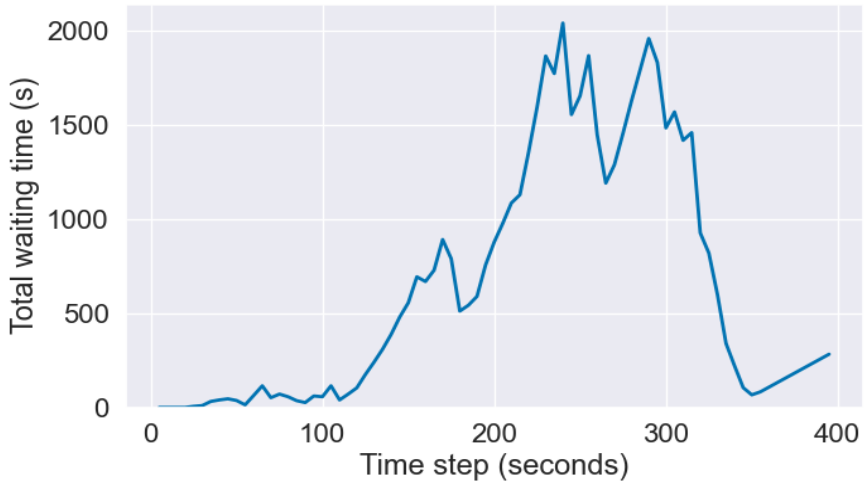


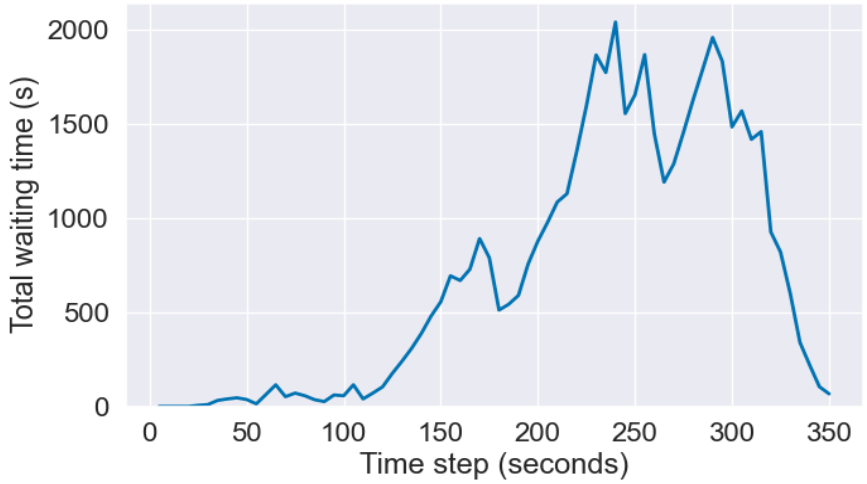


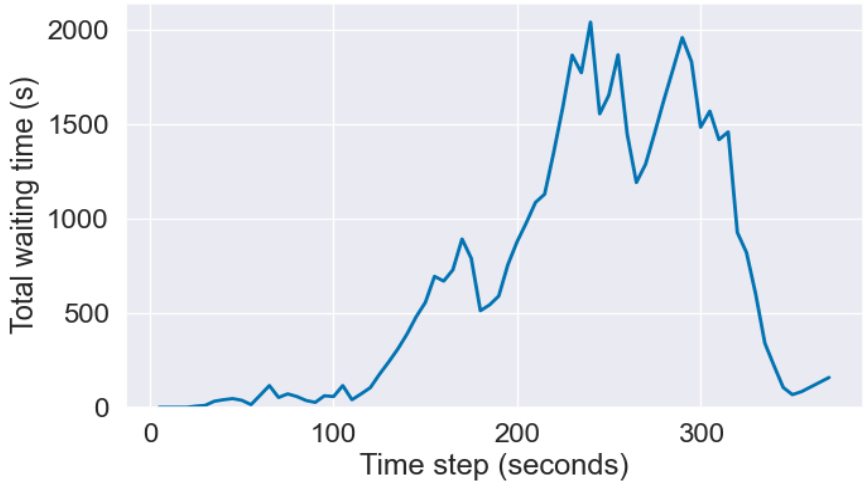


Sin dalla prima esecuzione, l'algoritmo Sarsa evidenzia le sue limitazioni con risultati scarsi. Fallisce nel completare gli episodi, mantenendo tempi di attesa elevati. Con il progredire degli episodi, non mostra miglioramenti significativi.

Seconda Run:

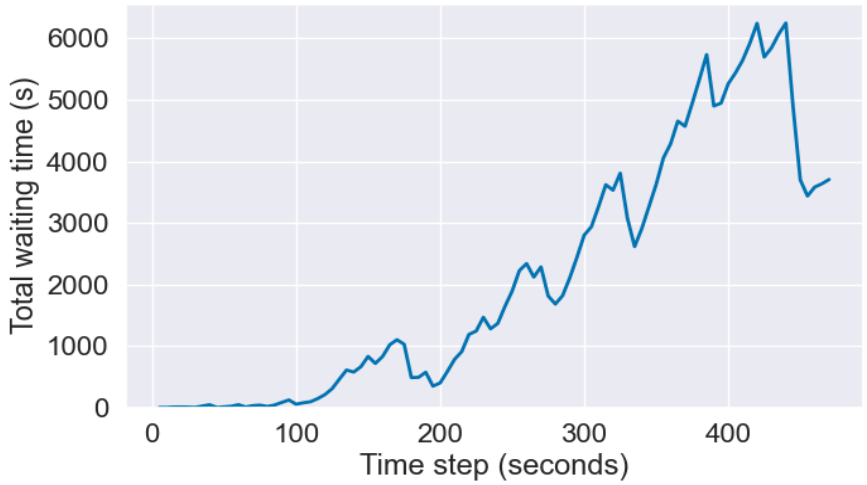


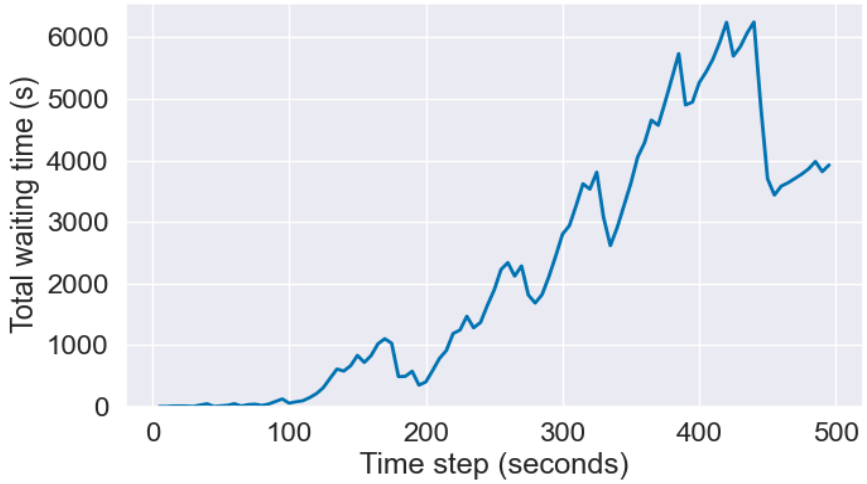


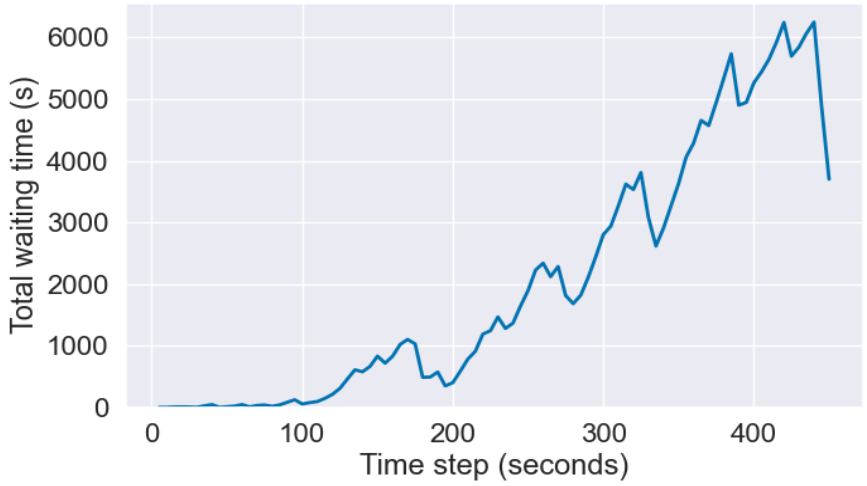


Nella seconda esecuzione, l'algoritmo Sarsa dimostra un miglioramento significativo nonostante il doppio dei veicoli coinvolti. Tuttavia, continua a fallire nel completare gli episodi e nel mantenere tempi di attesa accettabili.

Terza Run:







L'algoritmo SARSA mostra un aumento esponenziale dei tempi di attesa con l'aumentare dei veicoli, tuttavia, episodio dopo episodio, riesce a migliorare. È probabile che con un numero maggiore di episodi potrebbe ottenere miglioramenti significativi. Tuttavia, il progetto si è basato sul Q-Learning, utilizzando le stesse iterazioni e gli stessi episodi di quest'ultimo algoritmo.

6.4.3 Q-Learning VS SARSA

È stato effettuato un lavoro di fine tuning sia per il Q-Learning che per SARSA, tuttavia la differenza principale risiede nel fatto che, nel Q-Learning, anche con valori minori per tassi di apprendimento e diversi parametri, si otteneva un risultato soddisfacente e persino eccellente. Al contrario, nell'algoritmo SARSA, è stata la configurazione specifica dei parametri menzionati in precedenza a produrre i migliori risultati. Tentativi con parametri minori e differenti hanno invece portato a prestazioni scadenti e a tempi di attesa notevolmente prolungati

7. Conclusioni

In conclusione, il progetto di sviluppo di un sistema per il controllo del traffico urbano ha affrontato sfide significative durante l'implementazione, spostandosi da un'iniziale progettazione basata su Deep Q-learning verso l'utilizzo esclusivo dell'algoritmo Q-learning a causa di problemi di compatibilità con la libreria selezionata.
L'analisi accurata delle condizioni delle intersezioni stradali è stata fondamentale, focalizzandosi sulla densità del traffico e sulla presenza di veicoli in attesa. Sebbene l'integrazione delle condizioni meteorologiche fosse stata considerata inizialmente, vincoli nell'ambiente di lavoro hanno reso questa opzione impraticabile.
L'utilizzo della libreria sumo-rl per la configurazione dell'ambiente, la gestione delle osservazioni, delle azioni e delle ricompense ha rappresentato un punto focale nel processo di sviluppo.
In sintesi, il fine tuning sia per Q-Learning che per SARSA ha rivelato che, nel Q-Learning, anche con valori minori per tassi di apprendimento e parametri diversi, si ottiene un risultato eccellente. Al contrario, per SARSA, una configurazione specifica dei parametri ha portato ai risultati sufficienti, mentre tentativi con parametri minori e diversi hanno causato prestazioni scadenti e tempi di attesa prolungati.