



# Controllo Traffico con Apprendimento per Rinforzo

Applicazione dell'apprendimento per rinforzo alla gestione di un incrocio semaforico

23/02/2024

*di*

👤 Andrea Falbo  
a.falbo7@campus.unimib.it

👤 Ruben Tenderini  
r.tenderini@campus.unimib.it

Sotto la supervisione di  
Dr. Giuseppe Vizzari

## Contents

<b>1</b>	<b>Introduzione</b>	<b>1</b>
1.1	Definizione . . . . .	1
1.2	Obiettivo . . . . .	1
<b>2</b>	<b>Apprendimento per Rinforzo</b>	<b>2</b>
2.1	Probabilistic Planning . . . . .	2
2.2	Reinforcement Learning . . . . .	2
2.3	Q-Learning . . . . .	3
2.4	Deep Q-Learning . . . . .	3
2.5	SARSA Learning . . . . .	4
<b>3</b>	<b>Ambiente</b>	<b>5</b>
3.1	Introduzione . . . . .	5
3.2	MDP: Osservazioni, Azioni e Ricompense . . . . .	5
3.3	Configurazioni . . . . .	6
<b>4</b>	<b>Studio</b>	<b>8</b>
4.1	Traffico Basso - Traffico Basso . . . . .	8
4.2	Traffico Basso - Traffico Alto . . . . .	9
4.3	Traffico Alto - Traffico Basso . . . . .	9
4.4	Traffico Alto - Traffico Alto . . . . .	10
<b>5</b>	<b>Conclusioni</b>	<b>12</b>

## 1 Introduzione

### 1.1 Definizione

Si desidera confrontare l'efficacia di diverse strategie di apprendimento per rinforzo per la gestione del traffico di una intersezione semaforizzata, identificata come BI (Big Intersection). Nello specifico, saranno esaminate quattro diverse tecniche di apprendimento per rinforzo per la gestione del semaforo che verranno confrontate insieme alla strategia con ciclo fisso:

1. **Ciclo Fisso:** Le fasi del semaforo seguono un ciclo fisso, ripetendosi in modo prevedibile.
2. **Q-Learning:** L'operatività del semaforo è affidata a un agente addestrato mediante il metodo di Q-Learning.
3. **Deep Q-Learning:** L'operatività del semaforo è affidata a un agente addestrato mediante il metodo di Deep Q-Learning.
4. **Sarsa Learning:** L'operatività del semaforo è affidata a un agente addestrato mediante il metodo di Sarsa Learning.
5. **Sarsa Decay Learning:** L'operatività del semaforo è affidata a un agente addestrato mediante il metodo di Sarsa Learning il quale implementa un decadimento del fattore di esplorazione.

Ogni algoritmo verrà sperimentato attraverso 2 diverse configurazioni, variando gli iperparametri in modo da esplorare la sensibilità degli algoritmi e la loro adattabilità a contesti variabili, contribuendo a identificare parametri ottimali e migliorare la generalizzazione delle prestazioni nel controllo del traffico.

L'addestramento dei modelli considererà due diverse situazioni di traffico: traffico basso e traffico alto. Questa scelta è finalizzata a valutare la capacità dei modelli di adattarsi a varie dinamiche del traffico e di generalizzare efficacemente.

I modelli addestrati saranno successivamente valutati eseguendoli sia nella stessa situazione di traffico in cui sono stati addestrati, sia nella situazione non vista precedentemente. Tale metodologia mira a testare la capacità di generalizzazione dei modelli, evitando overfitting.

### 1.2 Obiettivo

L'obiettivo primario di questo studio è evidenziare se un approccio con apprendimento per rinforzo possa fornire prestazioni migliori rispetto ad un ciclo tradizionale dell'incrocio semaforico. Per farlo si cercheranno le configurazioni ottimali per ciascun algoritmo, per poi confrontare le prestazioni di questi attraverso diverse metriche e situazioni di traffico.

## 2 Apprendimento per Rinforzo

### 2.1 Probabilistic Planning

Nel contesto del Reinforcement Learning, l'agente affronta spesso la sfida di operare in ambienti in cui la conoscenza completa dello stato è impraticabile. Questo problema è affrontato efficacemente attraverso il Probabilistic Planning, una metodologia che incorpora l'incertezza nell'apprendimento dell'agente.

Il cuore di questo approccio risiede nella teoria delle probabilità, che consente all'agente di mantenere delle credenze probabilistiche, chiamate belief. Questi sono dinamici e variano in risposta alle nuove informazioni ottenute tramite i sensori.

Un concetto chiave in questo contesto è la Formula di Bayes, utilizzata per aggiornare i belief in base alle osservazioni sensoriali. Tale formula permette all'agente di ricalibrare le sue credenze in maniera razionale e guidata dai dati:

$$P(x|y_1 \wedge \dots \wedge y_n) = \frac{P(y_n|x)P(x|y_1 \wedge \dots \wedge y_{n-1})}{P(y_1 \wedge \dots \wedge y_n)}$$

Markov Localization è una tecnica particolarmente rilevante in questo contesto. Inizialmente, l'agente parte senza conoscenza dello stato attuale. Attraverso i sensori, percepisce parte dello stato corrente, consentendo l'applicazione della probabilità condizionata e l'aggiornamento dei belief tramite la Formula di Bayes. Questo processo si svolge anche quando l'agente compie azioni, percependo il cambiamento dell'ambiente e aggiornando di conseguenza i suoi valori.

Un concetto chiave associato a Markov è il Markov Decision Process (MDP). Questo modello, caratterizzato da un insieme discreto di stati  $S$ , un insieme di azioni  $A$ , una funzione di transizione  $T(s, a, s')$  e una funzione di ricompensa  $R(s, a, s')$ , fornisce una struttura formale per affrontare problemi decisionali in presenza di incertezza.

L'obiettivo principale in un MDP è massimizzare le ricompense, differenziandosi dai tradizionali problemi di ricerca per la sua enfasi sulla massimizzazione delle ricompense anziché sulla minimizzazione dei costi. L'approccio probabilistico adottato in Probabilistic Planning consente di modellare efficacemente l'incertezza, sviluppando strategie di pianificazione che si adattano dinamicamente alle condizioni mutevoli dell'ambiente.

### 2.2 Reinforcement Learning

Il Reinforcement Learning è una tecnica di apprendimento che prevede l'apprendimento di un agente attraverso l'interazione con un ambiente dinamico. L'agente interagisce con l'ambiente in modo sequenziale, compiendo azioni e ricevendo una ricompensa (*reward*).

Lo scopo dell'agente è quello di massimizzare la ricompensa cumulativa che viene fornita dall'ambiente in risposta alle sue azioni. L'RL si basa su un processo di apprendimento per esplorazione e sperimentazione, in cui l'agente deve scegliere le azioni da effettuare in modo da massimizzare la sua ricompensa. L'agente apprende dalla sua esperienza accumulando conoscenze e sviluppando strategie sempre più efficaci.

Lo scopo di un agente è quindi

$$\max \sum_{t=0}^T R(s_t, a_t)$$

dove  $T$  è il massimo degli istanti temporali.

Un agente che cerca di massimizzare la ricompensa cumulativa, potrebbe trovarsi in una situazione di indecisione nel caso di più sequenze di azioni la cui ricompensa totale sia uguale. Per risolvere questa indecisione si introduce il discount factor  $\gamma$  in modo da far scegliere all'agente la massimizzazione più veloce della

ricompensa cumulativa. La ricompensa al tempo  $t$  è data da:

$$R_t = \sum_{i=t}^T \gamma^{i-t} r_i$$

dove  $r_i$  è la ricompensa per la transizione dell'istante di tempo  $i$ . Questa sommatoria altro non è che la serie geometrica, e in quanto tale converge sempre a un valore finito anche per  $T = \infty$ .

### 2.3 Q-Learning

Il Q-Learning è un algoritmo specifico di Reinforcement Learning che si basa sulla costruzione di una tabella indicante il valore di ricompensa per ogni possibile stato al compimento di qualsiasi azione possibile. La procedura iterativa di costruzione della tabella coinvolge l'esplorazione dell'ambiente con azioni più o meno casuali.

Ad ogni passo, la tabella viene aggiornata con la seguente formula:

$$Q[s][a] = Q[s][a] + \alpha (r + \gamma \cdot \max(Q[s'])) - Q[s][a])$$

Dove:

- $Q(s, a)$  è la stima del valore d'azione per lo stato  $s$  e l'azione  $a$ ,
- $s$  è lo stato attuale,
- $a$  è l'azione compiuta,
- $r$  è la ricompensa ottenuta,
- $s'$  è lo stato successivo,
- $\gamma$  è il fattore di sconto,
- $\alpha$  è il tasso di apprendimento,
- $\max(Q[s'])$  restituisce la massima ricompensa ottenibile dallo stato  $s'$ .

La scelta dell'azione da compiere inizialmente è casuale fino a quando si decide di aver esplorato a sufficienza. La politica comunemente usata per questo scopo è la  $\epsilon$ -greedy, dove  $\epsilon$  rappresenta la probabilità di compiere un'azione casuale e viene ridotto progressivamente fino a un minimo.

Il Q-Learning è una tecnica potente ed efficace, capace di apprendere strategie di azione ottimali in diverse applicazioni. Tuttavia, può essere sensibile al rumore, all'indeterminazione delle azioni e alla gestione di ambienti continui. Inoltre, richiede una grande quantità di memoria per memorizzare la tabella, soprattutto quando l'ambiente ha uno spazio di stati  $S$  e di azioni  $A$  considerevole:  $(\theta(S \cdot A))$ .

### 2.4 Deep Q-Learning

Deep Q-learning è una versione avanzata di Q-learning che utilizza una rete neurale profonda per approssimare la funzione Q anziché una tabella. Ciò consente di affrontare problemi con spazi di stati e azioni molto più grandi e complessi. Il DQL è in grado di generalizzare meglio tra stati simili e può apprendere rappresentazioni più utili dei dati di input. Inoltre, l'uso di una rete neurale consente una maggiore flessibilità nell'apprendimento di strategie complesse.

Quindi, mentre Q-learning è limitato a problemi con spazi di stato e azione relativamente piccoli e richiede una tabella per memorizzare i valori Q, il Deep Q-learning supera questa limitazione utilizzando reti neurali profonde per approssimare la funzione Q, rendendolo adatto a problemi più complessi e di dimensioni maggiori.

## 2.5 SARSA Learning

Il SARSA (State-Action-Reward-State-Action) Learning è un algoritmo di apprendimento per il controllo di agenti in ambienti di apprendimento per rinforzo. L'algoritmo mantiene una funzione Q che stima il valore dell'azione in uno stato specifico. Durante l'apprendimento, l'agente esplora l'ambiente, seleziona azioni in base alla sua policy corrente e aggiorna la sua stima Q utilizzando l'equazione di aggiornamento SARSA:

$$Q(s, a) = Q(s, a) + \alpha [r + \gamma Q(s', a') - Q(s, a)]$$

Dove:

- $Q(s, a)$  è la stima del valore d'azione per lo stato  $s$  e l'azione  $a$ ,
- $\alpha$  è il tasso di apprendimento,
- $r$  è il reward ottenuto eseguendo l'azione  $a$  nello stato  $s$ ,
- $\gamma$  è il fattore di sconto,
- $s'$  è lo stato successivo,
- $a'$  è l'azione successiva selezionata dalla policy dell'agente.

Questa equazione regola la stima del valore d'azione in base al reward ottenuto, al valore d'azione successivo previsto e al tasso di apprendimento, guidando l'adattamento della policy dell'agente nel corso dell'apprendimento.

Mentre Q-Learning utilizza una strategia off-policy, apprendendo dalla massima stima del valore d'azione futura indipendentemente dall'azione effettivamente selezionata, SARSA adotta un approccio on-policy. Ciò significa che SARSA considera le azioni effettivamente scelte durante l'esplorazione, incorporando la policy corrente nella stima dei valori d'azione. Questa differenza fa sì che SARSA sia più sensibile alle politiche di esplorazione, garantendo che l'agente apprenda in base alle azioni che effettivamente compie.

### 3 Ambiente

#### 3.1 Introduzione

L'ambiente in cui gli agenti sono stati inseriti è caratterizzato da un incrocio con un singolo semaforo, noto come big-intersection.net.xml in Sumo-RL [1], e rinominato per il nostro studio in BI.net.xml. Questo ambiente definisce le corsie, i semafori, i sensi di marcia e altri aspetti. Ad ogni lato dell'incrocio sono presenti quattro corsie: una per svolte a sinistra, due per procedere diritto e una per procedere diritto o svoltare a destra.

Successivamente, questa configurazione è stata modellata in due scenari di traffico distinti, rappresentando livelli variabili di congestione stradale: traffico basso e traffico alto. Le configurazioni del traffico sono stati rinominati come

1. Traffico Bassa: BI\_50.rou.xml
2. Traffico Alta: BI\_150.rou.xml

Il numero di veicoli per le situazioni di traffico è stato determinato attraverso una serie di esperimenti iterativi per identificare condizioni ottimali. In conclusione, i valori che sono apparsi essere i più adeguati a rappresentare le 2 situazioni si sono rivelati:

1. Traffico Bassa: 50 veicoli per rotta
2. Traffico Alta: 150 veicoli per rotta

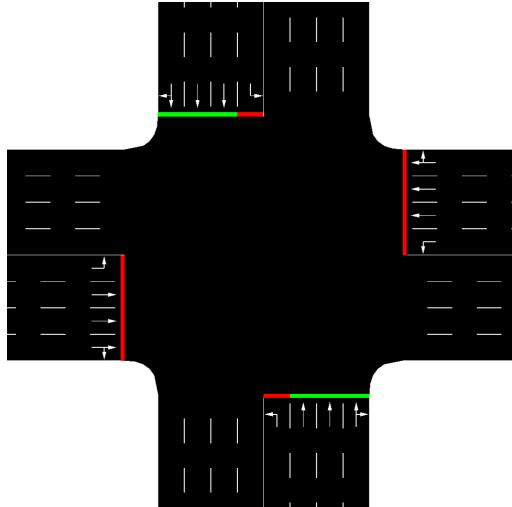


Figure 1: Rappresentazione dell'ambiente

#### 3.2 MDP: Osservazioni, Azioni e Ricompense

Il nostro modello, basato sul processo decisionale di Markov, rimarca le definizioni fornite da Sumo-RL:

L'osservazione predefinita per ciascun agente dei segnali stradali è un vettore:

$$obs = [phase\_one\_hot, min\_green, lane\_1\_density, \dots, lane\_n\_density, lane\_1\_queue, \dots, lane\_n\_queue]$$

dove:

- *phase-one-hot* è un vettore che codifica l'attuale fase verde attiva

- *min\_green* è un valore booleano che indica se sono già passati almeno *min\_green* secondi durante l'esecuzione
- *lane\_i\_density* è il numero di veicoli in arrivo nella corsia *i* diviso la capacità totale di tale corsia
- *lane\_i\_queue* è il numero di veicoli accodati in arrivo nella corsia *i* diviso la capacità totale di tale corsia

Lo spazio delle azioni è discreto. Ogni *delta\_time* secondi, ciascun agente semaforico ha la possibilità di selezionare la configurazione successiva della fase verde. Ogni volta che una fase termina, la successiva sarà preceduta da una fase gialla che dura *yellow\_time* secondi. Nel nostro caso avremo quindi  $|A| = 4$  azioni discrete.

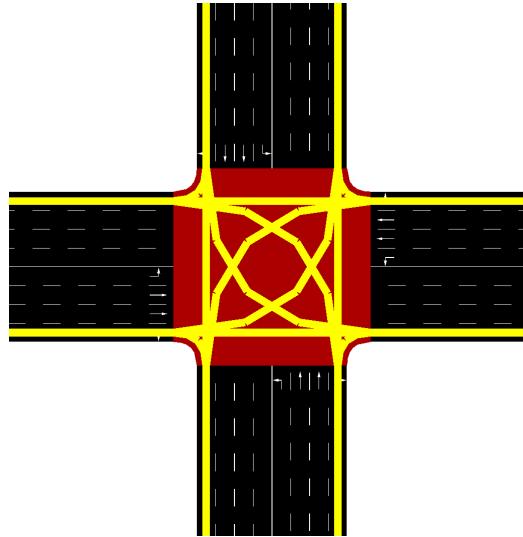


Figure 2: Rappresentazione delle azioni

La funzione di ricompensa, denominata Differential Waiting Time, è definita come il cambiamento cumulativo dei tempi di attesa dei veicoli:  $r_t = D_{\alpha_t} - D_{\alpha_{t+1}}$

Questa misura riflette quanto il tempo di attesa, in risposta a un'azione, sia migliorato o peggiorato. L'obiettivo è spingere l'agente a compiere azioni che portino a una diminuzione del tempo totale di attesa. Se questo diminuisce nello step successivo, la differenza sarà positiva, indicando un risultato favorevole.

### 3.3 Configurazioni

L'ambiente SUMO [2] è stato configurato attraverso i parametri definiti in Sumo-RL:

- Durata totale della simulazione: 100,000 secondi
- Intervallo tra le azioni dell'agente: 5 secondi
- Durata della fase gialla: 2 secondi
- Durata minima per una fase verde: 5 secondi
- Durata massima per una fase verde: 50 secondi

Per valutare le prestazioni degli agenti, sono state selezionate quattro metriche di sistema fornite da Sumo-RL:

- *system\_total\_stopped*: rappresenta il numero totale di veicoli fermi nel passo attuale.

- *system\_total\_waiting\_time*: indica la somma del tempo in secondi di simulazione in cui i veicoli sono stati fermi dall'ultima volta.
- *system\_mean\_waiting\_time*: rappresenta la media aritmetica di tutti i tempi di attesa dei veicoli.
- *system\_mean\_speed*: indica la media aritmetica delle velocità dei veicoli.

## 4 Studio

Per testare se gli algoritmi di apprendimento per rinforzo siano migliori nella gestione del traffico rispetto ad un semaforo a ciclo fisso, sono stati allenati i modelli sia in condizioni di traffico leggero che intenso, per poi testarli sia nelle stesse condizioni di allenamento che in condizioni diverse.

Per la valutazione sono state utilizzate le quattro metriche fornite da SUMO-RL. Nella presentazione verranno mostrati solamente i plot relativi al tempo d'attesa medio. I risultati ottenuti sono mostrati di seguito.

Per avere una visione migliore delle immagini, oppure per confrontare altre metriche, si consiglia di guardare la repository RL-Traffic-Control.

### 4.1 Traffico Basso - Traffico Basso

In questo esperimento, gli agenti sono stati allenati nell'ambiente a basso traffico, per poi essere stati testati sullo stesso ambiente.

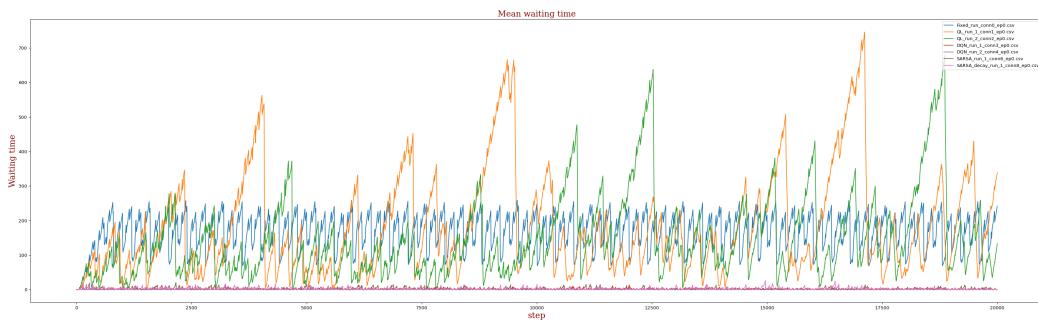


Figure 3: Tempo d'attesa medio: low traffic - low traffic

Si evidenzia come gli agenti Q-Learning sono peggiori e mostrano un'oscillazione maggiore rispetto al semaforo gestito con ciclo fisso, mentre gli agenti Sarsa e DQN offrono prestazioni notevolmente migliori rispetto al ciclo fisso.

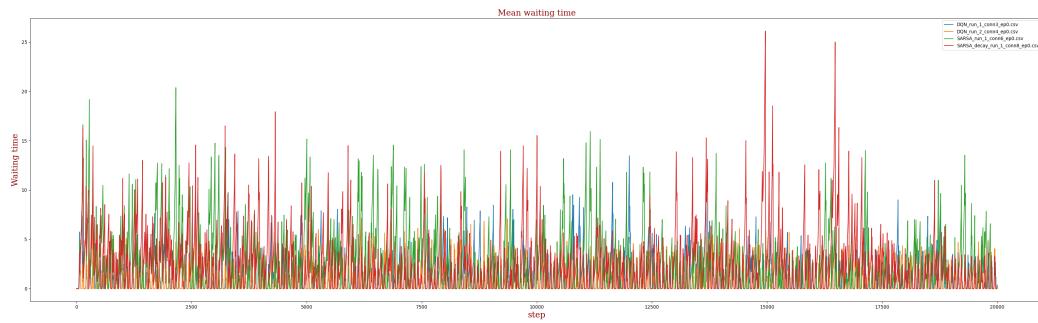


Figure 4: Tempo d'attesa medio DQN e SARSA: low traffic - low traffic

In questa visualizzazione sono stati rimossi gli agenti QL e FC per osservare meglio gli altri. Si nota come gli agenti DQN offrono prestazioni relativamente migliori rispetto agli agenti con strategia SARSA.

## 4.2 Traffico Basso - Traffico Alto

In questo esperimento, gli agenti sono stati allenati nell'ambiente a basso traffico, per poi essere testati sull'ambiente ad alto traffico.

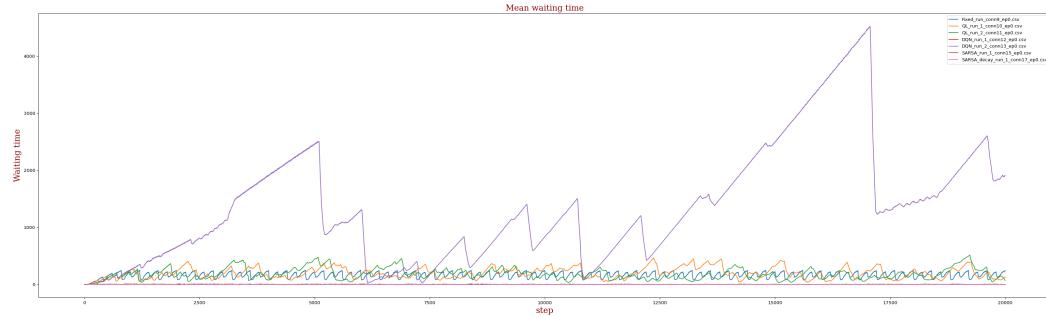


Figure 5: Tempo d'attesa medio: low traffic - high traffic

Si osserva velocemente come un agente DQN offre sorprendentemente prestazioni notevolmente peggiori rispetto al ciclo fisso che, come nel caso precedente, si comporta in maniera simile agli agenti QLearning. Di seguito un'altra immagine con visualizzazione migliore, escludendo dal plot gli agenti QLearning, l'agente fixed cycle e l'agente DQN con pessime prestazioni.

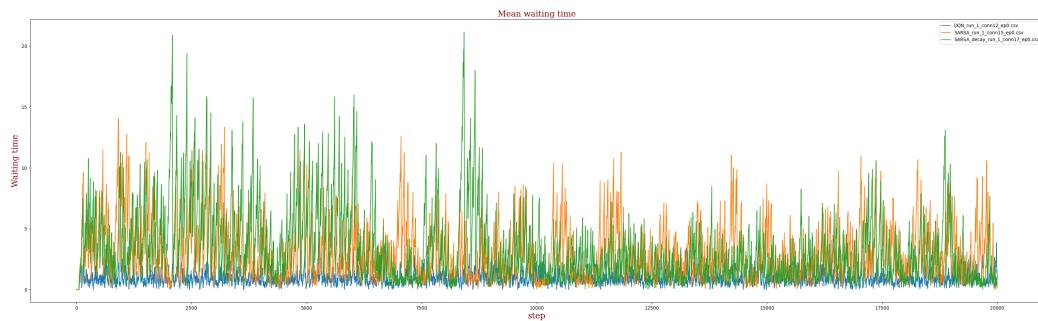


Figure 6: Tempo d'attesa medio Sarsa e DQN: low traffic - high traffic

Con questa nuova visualizzazione riusciamo a notare come l'agente DQN offre prestazioni notevolmente migliori rispetto ai SARSA.

## 4.3 Traffico Alto - Traffico Basso

In questo esperimento, gli agenti sono stati allenati nell'ambiente ad alto traffico, per poi essere testati sull'ambiente a basso traffico.

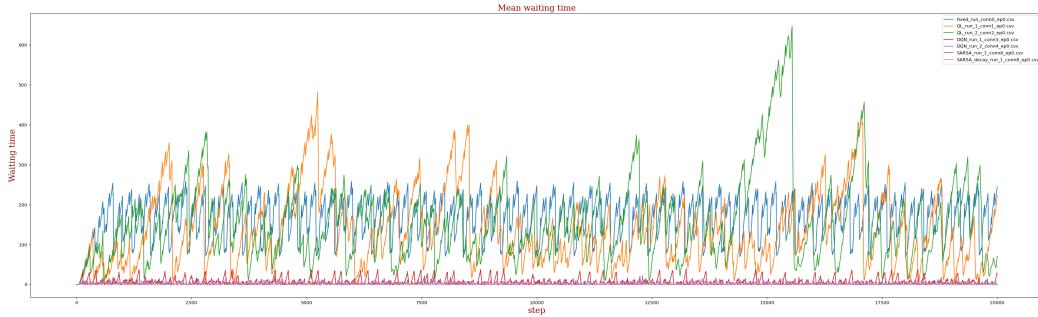


Figure 7: Tempo d'attesa medio: high traffic - low traffic

Come per il caso "low traffic - low traffic", gli agenti Q-Learning sono peggiori e mostrano un'oscillazione maggiore rispetto al semaforo gestito con ciclo fisso, mentre gli agenti Sarsa e DQN offrono prestazioni notevolmente migliori rispetto al ciclo fisso.

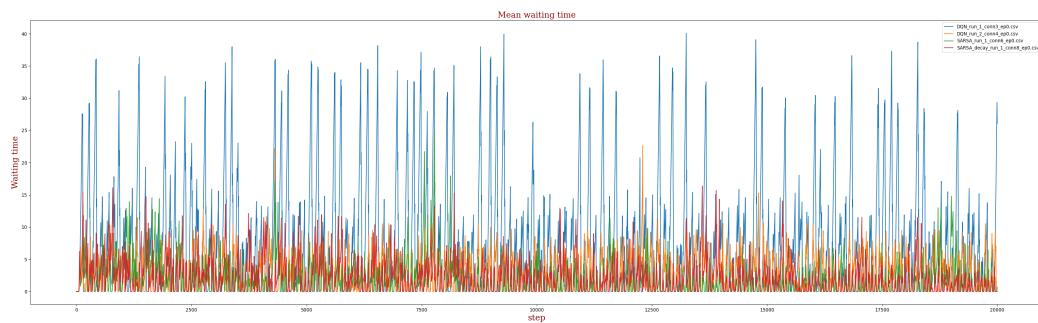


Figure 8: Tempo d'attesa medio DQN e SARSA: high traffic - low traffic

A differenza della visualizzazione su DQN e SARSA con addestramento sullo stesso scenario di traffico, qui gli agenti SARSA si comportano relativamente meglio rispetto ai DQN.

#### 4.4 Traffico Alto - Traffico Alto

In questo esperimento, gli agenti sono stati allenati nell'ambiente ad alto traffico, per poi essere stati testati sullo stesso ambiente.

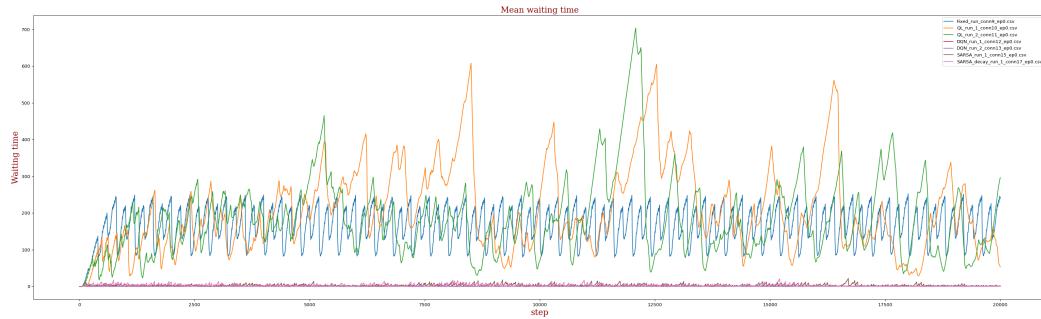


Figure 9: Tempo d'attesa medio: high traffic - high traffic

Come per il caso "low traffic - high traffic", gli agenti Q-Learning sono peggiori rispetto al ciclo fisso, mentre SARSA e DQN sono migliori, anche se non è chiaro chi è migliore tra questi. Di seguito una visualizzazione migliore.

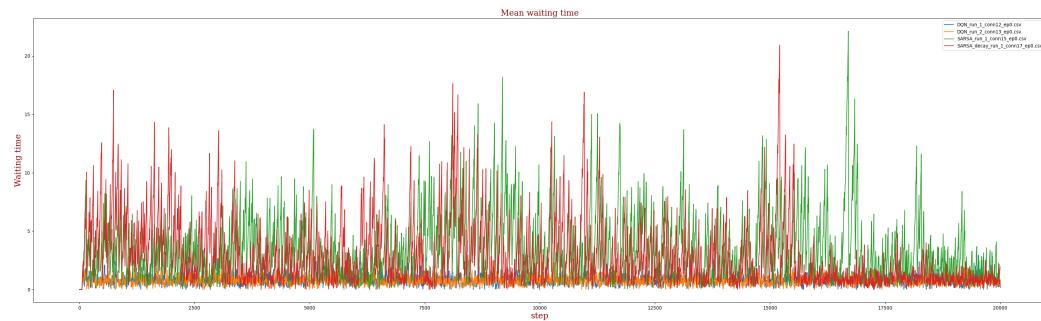


Figure 10: Tempo d'attesa medio Sarsa e DQN: high traffic - high traffic

In questo caso sia un agente DQN che un agente SARSA offrono prestazioni elevate. (arancione e blue, non molto visibili)

## 5 Conclusioni

In conclusione, i risultati del presente studio dimostrano chiaramente che gli algoritmi di apprendimento per rinforzo, come Deep Q-Learning e SARSA, sono in grado di migliorare significativamente le prestazioni nella gestione del semaforo rispetto alla tradizionale strategia di ciclo fisso per l'intersezione utilizzata. L'algoritmo Q-Learning, invece, ha mostrato spesso un peggioramento di prestazioni rispetto alla strategia di base.

Negli sviluppi successivi si potrebbe condurre uno studio migliore sugli iperparametri per verificare se anche QLearning possa risultare migliore rispetto al ciclo fisso. Inoltre, si potrebbero prendere in considerazione intersezioni con più agenti semaforici, in quanto più realistiche ad uno scenario reale.

## References

- [1] Lucas N. Alegre. SUMO-RL. <https://github.com/LucasAlegre/sumo-rl>, 2019.
- [2] DRL. Eclipse SUMO - Simulation of Urban MObility. <https://github.com/eclipse-sumo/sumo>, 2023.