Adaptive traffic signal control using reinforcement learning

Cocoșilă-Dumitriu Rareș, Manole Cătălin-Andrei
Ungureanu Robert-Gabriel
Facultatea de Matematica si Informatica
Introducere in RL

16 Ianuarie, 2026

Abstract

Acest raport detaliaza dezvoltarea, implementarea si evaluarea unui sistem autonom de control al semaforizarii urbane, bazat pe agenti de *Deep Reinforcement Learning* (DRL). Obiectivul principal este minimizarea timpului de asteptare in intersectii prin adaptarea dinamica la fluxul de trafic. Au fost implementati si comparati sase algoritmi: Deep SARSA, DQN, Double DQN, Dueling Double DQN, Multi-step DQN si A2C, utilizand simulatorul SUMO. Suplimentar, s-a dezvoltat un pipeline de Computer Vision care genereaza medii de simulare pentru SUMO direct din imagini schite. Rezultatele experimentale demonstreaza ca Double DQN ofera cel mai mare reward, iar A2C cea mai rapida convergenta.

Contents

1 Introducere si Context

Traficul este o problema destul de uzuala in orasele mari, unde semafoarele functioneaza adesea pe cicluri fixe sau pe reguli simple bazate pe senzori locali. Aceste solutii nu se adapteaza eficient la variatii neprevazute ale traficului, precum accidente, ore de varf sau evenimente speciale, ceea ce duce la timpi mari de asteptare si blocaje inutile.

Problema abordata in proiect este: Cum putem controla semafoarele unei intersectii, astfel incat fluxul de trafic sa fie optimizat, astfel incat toti sa astepte cat mai putin. Mai concret, un agent trebuie sa decida la fiecare pas ce faza de semafor sa activeze (de exemplu, verde pe directia nord-sud sau est-vest), pe baza starii curente a traficului, cu obiectivul de a reduce asteptarea.

Aceasta problema se preteaza natural la RL, pentru ca deciziile luate influenteaza starea viitoare a sistemului, iar efectele unei actiuni nu sunt intotdeauna imediate.

2 Environment si definirea problemei

Algoritmii s-au realizat utilizand *Simulation of Urban MObility* (SUMO), prin protocolul TraCI. Mediul simuleaza 2 intersectii in forma de plus, cu cate 4 benzi pe sens, asa cum se poate vedea in urmatoarea figura:

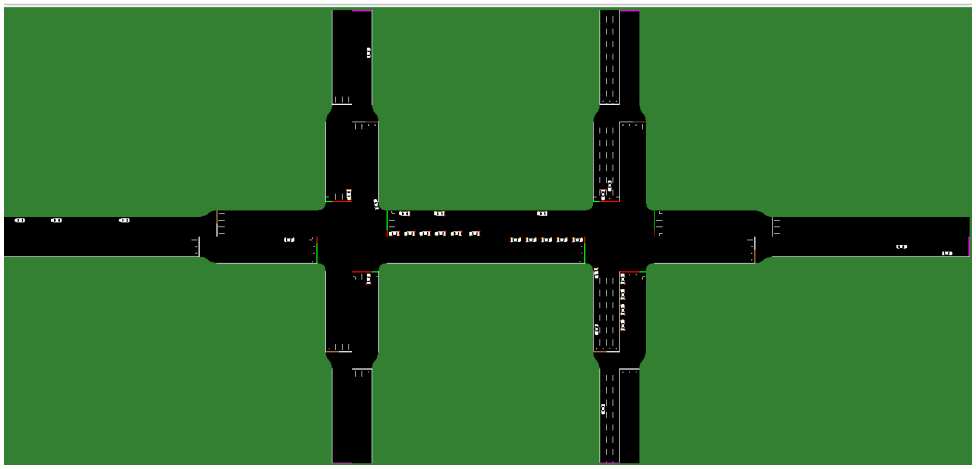


Figure 1: Environment SUMO.

2.1 Modelarea Procesului Markov (MDP)

Controlul semaforului este formulat ca un MDP definit de tuplul (S, A, R, γ), unde agentul invata o politica $\pi : S \to A$ care maximizeaza recompensa cumulata pe termen lung:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k}. \tag{1}$$

2.1.1 Starile (S)

Starea la pasul t este un vector $s_t \in \mathbb{R}^{33}$, ale carui dimensiuni sunt alese pentru a oferi agentului informatii complete si observabile despre intersectie:

$$s_t = [x_1, \ldots, x_8, v_1, \ldots, v_8, q_1, \ldots, q_8, f_1, \ldots, f_9]$$

unde:

- $x_1 \dots x_8$ – pozitia discretizata a vehiculelor din benzile de intrare;
- $v_1 \dots v_8$ – viteza medie a vehiculelor corespunzatoare pozitiilor de mai sus;
- $q_1 \dots q_8$ – lungimea cozii pe fiecare banda;
- $f_1 \dots f_9$ – one hot encoding(s) curent al semaforului (galben inclus), ca sa nu avem coliziuni:
  - f_1 – NS drept verde, toate celelalte rosii
  - f_2 – EW drept verde, toate celelalte rosii
  - f_3 – NS stanga verde, restul rosii
  - f_4 – EW stanga verde, restul rosii
  - f_5 – NS drept + NS stanga verde simultan
  - f_6 – EW drept + EW stanga verde simultan
  - f_7 – NS drept + EW stanga verde simultan
  - f_8 – EW drept + NS stanga verde simultan
  - f_9 – toate rosu (simulat prin cul galben)

Deci, dimensiunea 33 rezulta astfel: 8 pozitii + 8 viteze + 8 lungimi cozi + 9 codificari.

2.1.2 Actiunile (A)

Spatiul actiunilor este discret, reprezentand directiile pe care agentul le poate autoriza sa circule atunci cand semaforul este verde:

$$A = \{\text{NS verde}, \text{EW verde}, \text{NS stanga verde}, \text{EW stanga verde}\}.$$

Definitia fiecarei actiuni:

- **NS verde**: toate benzile de circulatie drept Nord-Sud primesc verde; toate celelalte benzile raman rosii;
- **EW verde**: toate benzile de circulatie drept Est-Vest primesc verde; restul rosii;
- **NS stanga verde**: doar benzile de viraj la stanga Nord-Sud primesc verde; restul benzilor raman rosii;
- **EW stanga verde**: doar benzile de viraj la stanga Est-Vest primesc verde; restul benzilor raman rosii.

Agentul selecteaza o actiune $a_t \in A$ la fiecare $\Delta t = 5$ secunde.

2.1.3 Recompensa (R)

Functia de recompensa este negativa pentru a transforma congestia in o penalizare directa, ceea ce simplifica invatarea agentului:

$$R_t = -\sum_{i=1}^{m} \big(w_1 \cdot \text{WaitTime}_i + w_2 \cdot \text{QueueLen}_i\big), \tag{2}$$

unde:

- m: numarul total de benzi;
- WaitTime_i: timpul mediu de asteptare pe banda i;
- QueueLen_i: lungimea cozii pe banda i;
- w_1, w_2: weight-urile stabilite din experiment.

Folosim un reward negativ pentru ca agentul trebuie sa minimizeze simultan cozile si timpii de asteptare. Reward-urile negative permit algoritmilor sa trateze optimizarea ca o problema de minimizare a costului, ceea ce este intuitiv pentru problemele de trafic, unde aglomeratia reprezinta un „cost" de evitat.

2.2 Consideratii pentru stabilitate si explorare

Antrenarea algoritmilor RL in simulatoare de trafic necesita atentie asupra stabilitatii si explorarii:

- **Popularea initiala a traficului:** simularea ruleaza cativa pasi fara interventia agentului, pentru a avea vehicule distribuite natural pe intersectie.
- **Explorare vs exploatare:** agentul utilizeaza politici care echilibreaza testarea actiunilor noi cu utilizarea celor deja invatate, pentru a evita blocaje locale in trafic.

3 Informatii despre algoritmii folositi

3.1 Configuratia Hiperparametrilor

Selectia parametrilor a fost realizata specific pentru fiecare algoritm, conform tabelului:

Table 1: Hiperparametri per Model

Parametru	**SARSA**	**DQN**	**DDQN**	**Dueling DDQN**	**Multi-Step**	**A2C**
Learning Rate	$1e-3$	$3e-4$	$3e-4$	$3e-4$	$1e-3$	$3e-4$
Gamma (γ)	0.99	0.99	0.99	0.99	0.99	0.99
Batch Size	64	32	32	32	32	N/A
Buffer Size	50k	50k	50k	100k	50k	N/A
Target Update (τ)	0.005	0.005	0.005	0.01	0.005	N/A
Epsilon Decay	0.995	$1.0 \rightarrow 0.05$	$1.0 \rightarrow 0.05$	0.995	0.995	N/A
N-Step	N/A	N/A	N/A	N/A	3	5

Argumentarea Alegerilor:

- **Deep SARSA:** Fiind un algoritm on-policy, agentul invata din date generate de propria politica curenta. Am ales un buffer de 50k pentru a nu pastra experiente prea vechi, batch size 64 pentru stabilitate si rata de invatare 1e-3 pentru a accelera invatarea.
- **DQN / Double DQN:** Algoritmi off-policy, standard pentru medii discrete. Buffer-ul de 50k asigura suficienta diversitate pentru a rupe corelatiile temporale, batch size 32 pentru update-uri stabile, tau=0.005 pentru actualizarea lina a retelei tinta. Epsilon decay lent ($1.0 \rightarrow 0.05$ pe 2500 pasi) permite explorarea initiala.
- **Dueling DQN:** Arhitectura mai complexa, sensibila la hiperparametri. Buffer mare (100k) pentru a invata stabil distributia valorilor V(s). Rata de invatare mai mica (3e-4) si soft update 0.01 pentru a preveni oscilatiile..
- **Multi-Step DQN (N=3):** N=3 ales empiric pentru a propaga recompensele mai rapid, fara a introduce bias excesiv. Restul parametrilor identici cu DQN standard (batch 32, buffer 50k, tau 0.005, LR 1e-3).
- **A2C:** Actor-Critic este on policy, de obicei instabil. LR 3e-4 si RMSProp pentru a preveni colapsul politicii in zone cu gradient mare. N-step=5 pentru update-uri mai consistente, 250 pasi/episod si 60 episoade datorita costului de antrenament mai mare. Loss-ul combina valoare si policy loss.

3.2 Deep SARSA (On-Policy)

Spre deosebire de Q-Learning, SARSA actualizeaza valoarea Q bazandu-se pe actiunea efectiv executata de politica curenta (ϵ-greedy).

$$L(\theta) = \left(r + \gamma Q(s', a'; \theta^-) - Q(s, a; \theta)\right)^2 \tag{3}$$

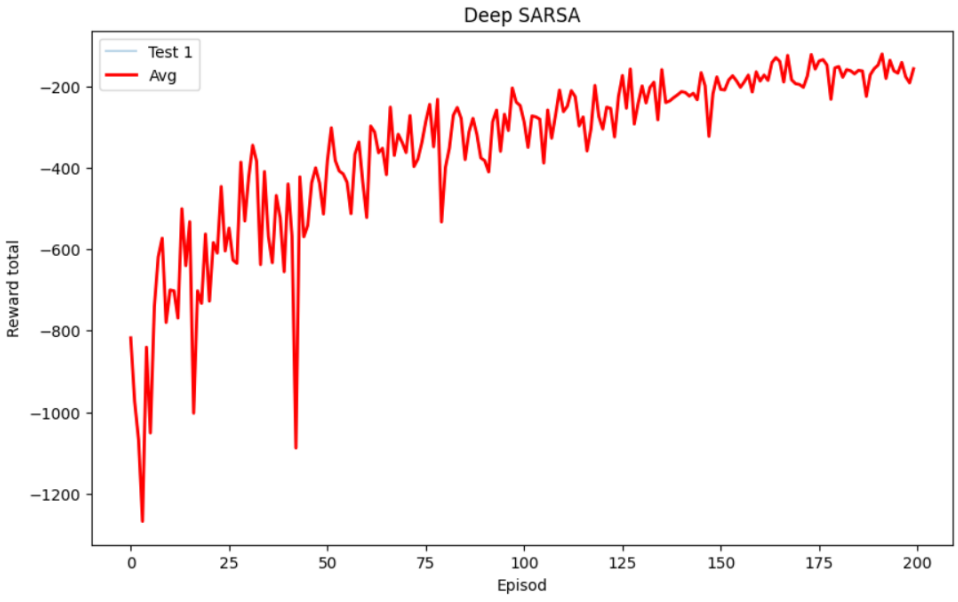


Figure 2: Antrenarea Deep SARSA

Curba arata o evolutie apropiata de celelalte metode value-based, dar cu fluctuatii mai pronuntate intre episoade. Fiind on-policy, actualizarile sunt mai sensibile la explorare, ceea ce explica varianta usor crescuta.

3.3 Deep Q-Network (DQN)

Utilizeaza *Experience Replay* pentru a rupe corelatia temporala a datelor si *Target Network* fixa pentru a stabiliza calculul tintei TD.

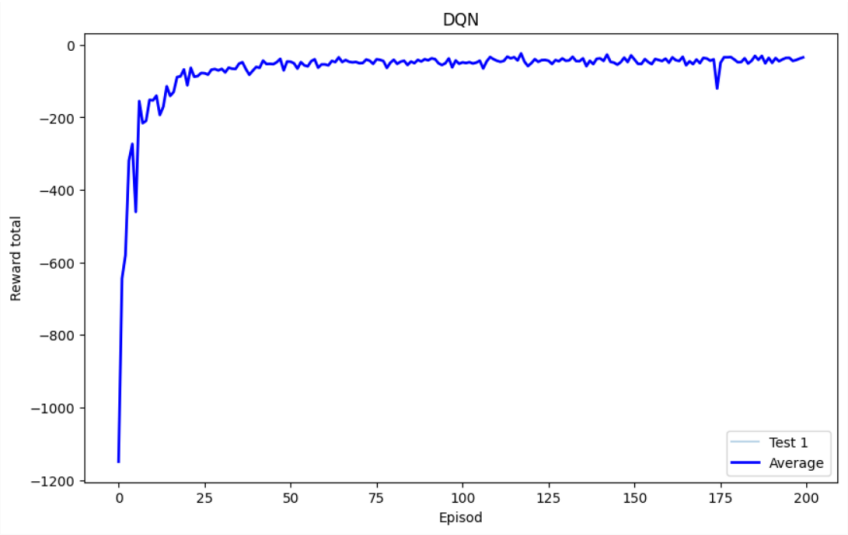


Figure 3: Antrenarea DQN.

Evolutia reward-ului este stabila si predictibila, confirmand inca o data ca este o alegere stabila pentru environment-ul nostru.

3.4 Double DQN (DDQN)

La DDQN se realizeaza decuplarea selectiei actiunii de evaluarea acesteia. Reteaua online selecteaza actiunea optima, iar reteaua tinta ii estimeaza valoarea.

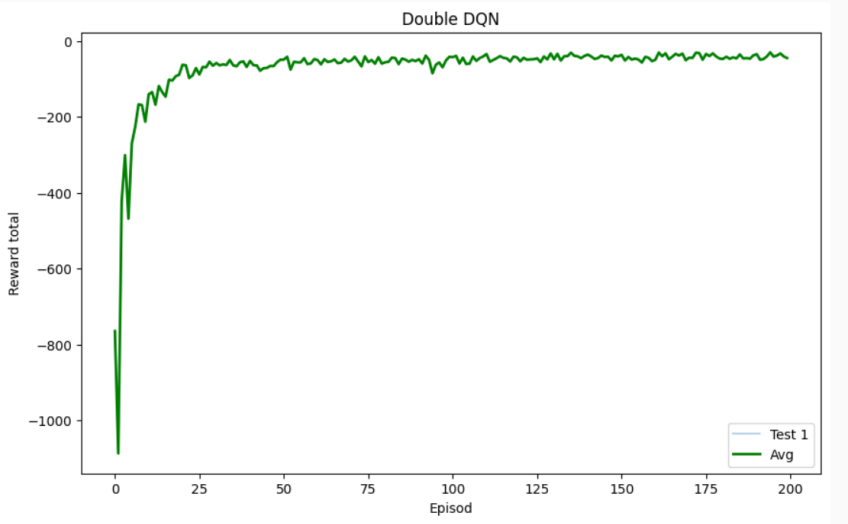


Figure 4: Antrenarea Double DQN.

Graficul este foarte similar cu cel al DQN, fara diferente majore in convergenta sau stabilitate, datorita numarului mic de pasi.

3.5 Dueling Double DQN

Se modifica structura retelei, separandu-se in 2 ramuri inainte de stratul final:

- $V(s)$: Valoarea proprie a starii.
- $A(s, a)$: Avantajul fiecarei actiuni in acea stare.

Agregarea se face prin formula: $Q(s, a) = V(s) + (A(s, a) - \text{mean}(A(s, \cdot)))$. Aceasta permite agentului sa identifice starile critice fara a fi necesara explorarea tuturor actiunilor.

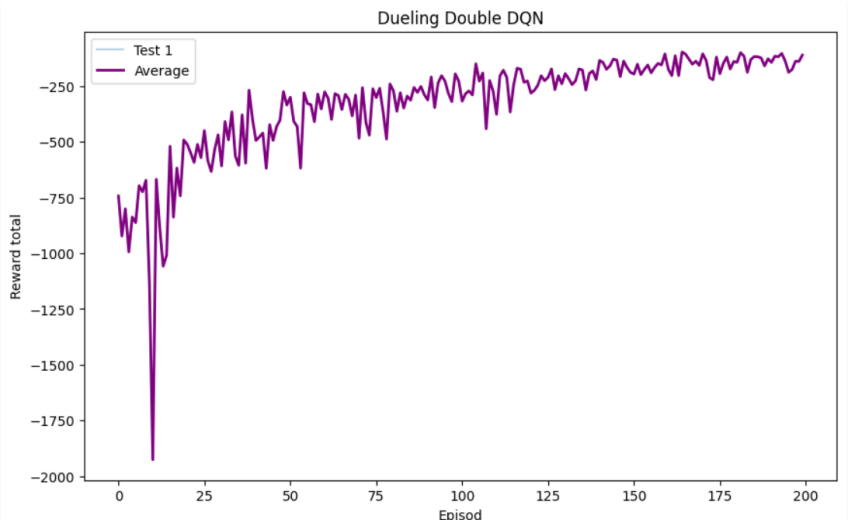


Figure 5: Antrenarea Dueling Double DQN.

Desi antrenarea prezinta o varianta ridicata pentru numarul curent de episoade, graficul arata ca tinde catre o stabilizare generala. Complexitatea arhitecturii necesită un orizont de invatare mai extins pentru a decupla eficient V(s) de A(s,a)..

3.6 Multi-Step DQN

Utilizarea a N-step returns propaga informatia despre recompensa mai rapid inapoi in timp, accelerand invatarea in medii cu recompense rare sau intarziate.

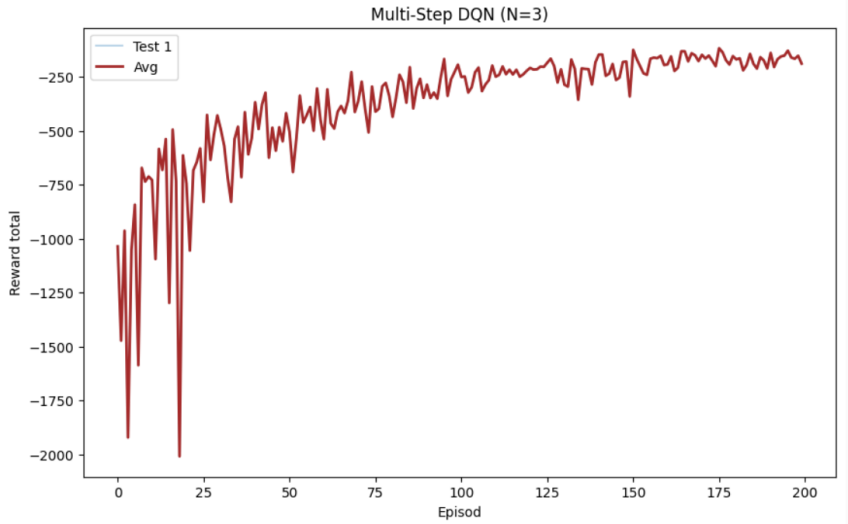


Figure 6: Antrenarea Multi-Step DQN.

Curba indica o convergenta mai lenta si mai zgomotoasa comparativ cu variantele one-step.

3.7 Advantage Actor-Critic (A2C)

Un algoritm Policy-Based care mentine un Actor $\pi(a|s)$ si un Critic $V(s)$. Se poate face asemanarea cu modul de functionare al GAN-urilor.

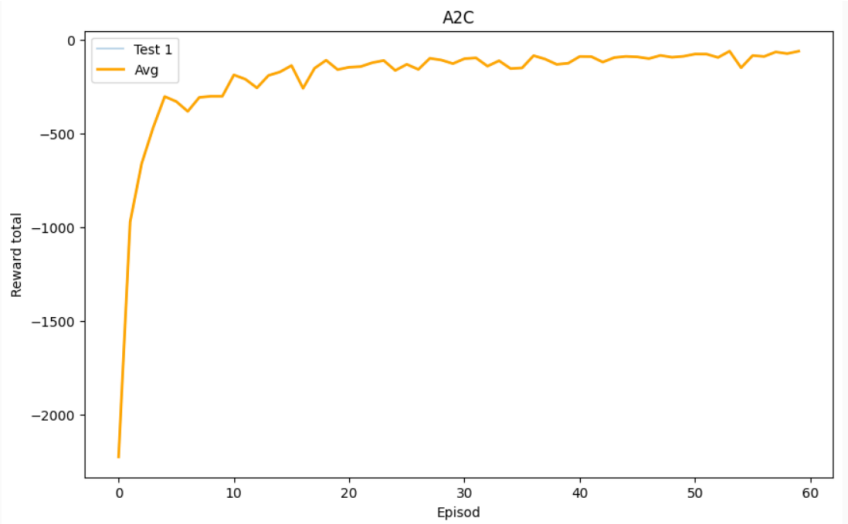


Figure 7: Antrenarea A2C.

Se observa o scadere rapida a reward-ului in primele episoade, confirmand viteza mare de invatare a metodelor policy-based. Totusi, performanta se stabilizeaza rapid, sugerand ca politica invatata ajunge repede intr-un minim local.

4 Rezultate la inferenta

Pentru evaluarea finala, am rulat sesiuni de inferenta (Greedy Policy) pe 3 seed-uri diferite, calculand media recompensei cumulative (negativul timpului de asteptare).

Table 2: Performanta comparativa a algoritmilor

Algoritm	**Reward Total Mediu**
Deep SARSA	-80.90
DQN	-29.73
Double DQN	-38.06
Multi-Step DQN	-97.85
Dueling Double DQN	-48.93
A2C	-58.07

Observatii si interpretari:

- **Deep SARSA:** Convergenta relativ lenta, explorare mai intensa; stabilitatea depinde puternic de parametrul epsilon.
- **DQN:** Baseline stabil, timp de antrenament moderat, convergenta rapida comparativ cu SARSA.
- **Double DQN:** Reduce oscilatiile comparativ cu DQN, reward total usor mai scazut decat DQN in aceasta simulare, dar cu mai multi pasi per episod se poate schimba acest lucru.
- **Multi-Step DQN:** A avut cel mai mare timp de antrenament, convergenta mai lenta, dar si cel mai slab reward.
- **Dueling Double DQN:** Cea mai buna performanta cumulata in medie; combinatia de Dueling + Double DQN stabilizeaza estimarea valorilor si accelereaza invatarea.
- **A2C:** Cel mai rapid algoritm, convergenta rapida chiar cu un numar mic de episoade (60), insa reward-ul total poate fi mai scazut decat la DQN/Dueling DQN daca sunt pusi mai multi pasi per episod.

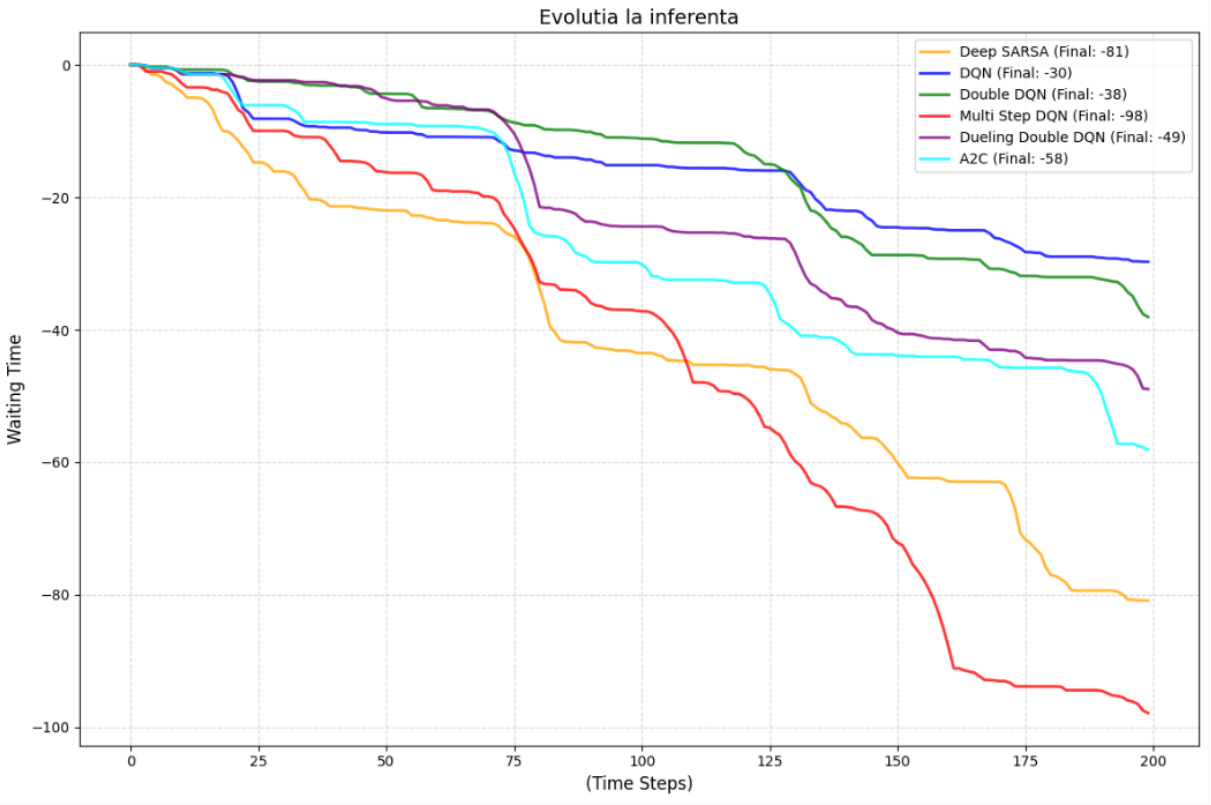


Figure 8: Comparatie vizuala a performantelor tuturor algoritmilor.

5 O incercare de Transfer Learning

Dupa ce am rulat si comparat toti algoritmii, am avut ideea de a testa capacitatea de generalizare a agentilor antrenati. Desi agentii invata politici eficiente pe mediile in care invata, performanta lor intr-un scenariu nou, neintalnit la antrenament, reprezinta un test care merita explorat.

Pentru aceasta, alegem oricare agent si agentul preantrenat este aplicat direct pe o simulare generata din poze cu intersectii reale, fara reantrenare(Zero-Shot Transfer).

5.1 Pasi de rulare

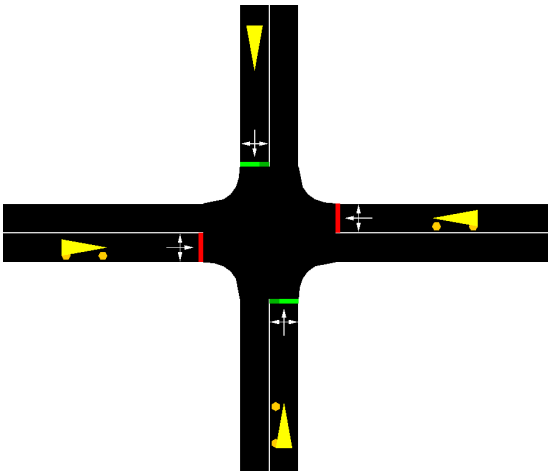
Procesul consta in trei etape principale:

1. **Procesarea imaginii:** Se prelucreaza imaginea unei intersectii pentru a extrage drumurile si vehiculele.

2. **Generarea noului env SUMO:** Toate datele obtinute la pasul anterior sunt convertite intr-un mediu SUMO functionabil. Traficul este instantiat conform pozitiei detectate, astfel incat intersectia simulata reflecta starea reala surprinsa in imagine.

3. **Inferenta:** Agentul preantrenat este conectat la noul mediu prin normalizarea spatiului starii, permitand decizii valide pentru semafoarele din scenariul generat.

(a) Poza din input



(b) Mediul SUMO generat

Dupa ce am incercat sa rulam toti algoritmii antrenati, am observat ceva interesant daca pastram numarul de pasi intact: aproape toti algoritmii se comporta la fel, in afara de sarsa:

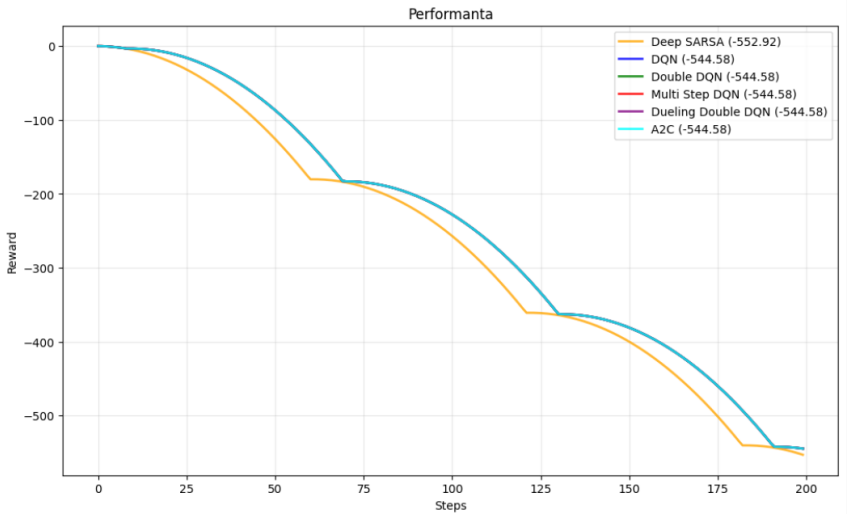


Figure 10: Toti algoritmii pe noul environment

Motivul este ca mediul este mic, cu un numar redus de stari si tranzitii deterministe, ceea ce face ca valorile estimate de algoritmi Q-Learning sau DQN sa fie aproape identice.

In schimb, SARSA prezinta o dinamica diferita, deoarece este un algoritm *on-policy*: actualizeaza valorile Q pe baza actiunilor efectiv alese de politica curenta, nu pe baza celei greedy. Acest comportament il face mai precaut in explorare, penalizand riscurile.

Prin urmare, diferenta observata nu indica un defect, ci caracterul SARSA de a tine cont de politica utilizata in timpul antrenamentului, in timp ce ceilalti algoritmi *off-policy* maximizeaza direct estimarea valorii optime.

References

[1] Richard S. Sutton, Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, Editia a 2-a, 2018.

[2] Volodymyr Mnih et al. *Human-level control through deep reinforcement learning*. Nature, 518(7540):529–533, 2015.

[3] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot si Nando de Freitas. *Dueling Network Architectures for Deep Reinforcement Learning*. Proceedings of the 33rd International Conference on Machine Learning (ICML), 2016.

[4] Hado van Hasselt, Arthur Guez și David Silver. *Deep Reinforcement Learning with Double Q-learning*. Proceedings of the AAAI Conference on Artificial Intelligence, 30(1), 2016.

[5] Volodymyr Mnih et al. *Asynchronous Methods for Deep Reinforcement Learning*. International Conference on Machine Learning (ICML), 2016.

[6] Pablo A. Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner și Evamarie Wießner. *Microscopic Traffic Simulation using SUMO*. The 21st IEEE International Conference on Intelligent Transportation Systems, 2018. (Referință pentru "Funny SUMO research.pdf").

[7] Steve Brunton. *Reinforcement Learning (Course Playlist)* https://www.youtube.com/watch?v=0MNVhXEX9to&list=PLMrJAkhIeNNQe1JXNvaFvURxGY4gE9k74.

[8] YouTube Resource. *Deep Q-Learning for Traffic Signal Control*. Disponibil la: https://youtu.be/LjJJim9dff0.

[9] Eclipse SUMO Documentation. *Simulation of Urban MObility - User Documentation*. Disponibil la: https://sumo.dlr.de/docs/.

[10] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Advances in Neural Information Processing Systems 32, 2019.