UNIVERZITET U SARAJEVU
ELEKTROTEHNIČKI FAKULTET
ODSJEK ZA RAČUNARSTVO I INFORMATIKU

Adaptivno upravljanje saobraćajnom signalizacijom

ZAVRŠNI RAD

- PRVI CIKLUS STUDIJA -

Student:
Harun Goralija

Mentor:
Doc. dr Amra Delić Halilović

Sarajevo,
septembar 2025

Univerzitet u Sarajevu
Elektrotehnički fakultet
Odsjek za Računarstvo i informatiku

Izjava o autentičnosti radova

Završni rad
I ciklusa studija

Ime i prezime: Harun Goralija
Naslov rada: Adaptivno upravljanje saobraćajnom signalizacijom
Vrsta rada: Završni rad Prvog ciklusa studija
Broj stranica: 38

Potvrđujem:

- da sam pročitao dokumente koji se odnose na plagijarizam, kako je to definirano Statutom Univerziteta u Sarajevu, Etičkim kodeksom Univerziteta u Sarajevu i pravilima studiranja koja se odnose na I i II ciklus studija, integrirani studijski program I i II ciklusa i III ciklus studija na Univerzitetu u Sarajevu, kao i uputama o plagijarizmu navedenim na web stranici Univerziteta u Sarajevu;
- da sam svjestan univerzitetskih disciplinskih pravila koja se tiču plagijarizma;
- da je rad koji predajem potpuno moj, samostalni rad, osim u dijelovima gdje je to naznačeno;
- da rad nije predat, u cjelini ili djelimično, za stjecanje zvanja na Univerzitetu u Sarajevu ili nekoj drugoj visokoškolskoj ustanovi;
- da sam jasno naznačio prisustvo citiranog ili parafraziranog materijala i da sam se referirao na sve izvore;
- da sam dosljedno naveo korištene i citirane izvore ili bibliografiju po nekom od preporučenih stilova citiranja, sa navođenjem potpune reference koja obuhvata potpuni bibliografski opis korištenog i citiranog izvora;
- da sam odgovarajuće naznačio svaku pomoć koju sam dobio pored pomoći mentora i akademskih tutora/ica.

Sarajevo, septembar 2025

Potpis:

_____

Harun Goralija

Elektrotehnički fakultet, Univerzitet u Sarajevu
Odsjek za Računarstvo i informatiku.
Doc. dr Amra Delić Halilović, dipl.el.ing
Sarajevo, septembar 2025

Postavka zadatka završnog rada I ciklusa:
Adaptivno upravljanje saobraćajnom signalizacijom

Cilj ovog rada je implementacija adaptivnog upravljanja saobraćajnom signalizacijom korištenjem jednostavnog modela reinforcement learninga (RL) i realnih saobraćajnih podataka. Klasični pristupi upravljanju semaforima često koriste fiksne vremenske intervale ili unaprijed definisane planove, dok RL omogućava dinamičku prilagodbu signalizacije na osnovu trenutnog saobraćajnog stanja. Kako bi se analizirala efikasnost predloženog modela, potrebno je simulirati saobraćaj u scenarijima sa i bez adaptivnog upravljanja koristeći postojeće saobraćajne simulatore. Evaluacija modela će se provesti primjenom standardnih metrika za procjenu saobraćajne prohodnosti i zagušenja, čime će se kvantitativno analizirati potencijalne prednosti adaptivnog sistema u odnosu na klasične metode upravljanja.

Mentor: Doc. dr Amra Delić Halilović

Polazna literatura:

[1] Abdulhai, B., Pringle, R., Karakoulas, G. J., "Reinforcement learning for true adaptive traffic signal control", Journal of Transportation Engineering, Vol. 129, No. 3, 2003, str.278–285.
[2] Yau, K.-L. A., Qadir, J., Khoo, H. L., Ling, M. H., Komisarczuk, P., "A survey on reinforcement learning models and algorithms for traffic signal control", ACM Computing Surveys (CSUR), Vol. 50, No. 3, 2017, str. 1–38.
[3] Balaji, P., German, X., Srinivasan, D., "Urban traffic signal control using reinforcement learning agents", IET Intelligent Transport Systems, Vol. 4, No. 3, 2010, str. 177–188.

Doc. dr Amra Delić Halilović, dipl. ing. el.

Sadržaj

Popis slika

Popis tabela

Poglavlje 1

Uvod

U ovom poglavlju će biti predstavljene te pojašnjene sljedeće stavke:

- Opis problema i motivacija
- Sažetak predloženog rješenja, ciljevi rada i metodologija
- Sažetak rezultata
- Struktura rada

1.1 Opis problema i motivacija

Problem upravljanja saobraćajnom signalizacijom postoji gotovo jednako dugo kao i samo upravljanje saobraćajem uz pomoć ovog metoda. Razlog tome jeste što je saobraćajna signalizacija nastala kao pokušaj rješenja ubrzanog povećanja broja putničkih vozila, te je samim tim postajalo sve manje efikasno koristiti tada konvencionalne načine upravljanja saobraćajem, poput pravila desne strane u raskrsnicama.

Međutim, iako je saobraćajna signalizacija uspjela da riješi problem jedne određene količine vozila, broj istih je nastavio da raste, te je čovječanstvo nastavilo da razvija efikasnije, svojevremeno optimalne načine da se organizuju ukrštanja saobraćajnica.

Uprkos tome, sve raskrsnice koje su do tada organizirane uz pomoć svjetlosne signalizacije, bivaju suočene sa dvije mogućnosti. Prva je da se uradi njena potpuna rekonstrukcija, što zahtijeva veliku količinu ulaganja, te preusmjeravanje saobraćaja na druge saobraćajnice, usljed radova. Druga mogućnost jeste da se rad semafora izmijeni, na način da njihovi intervali nisu predodređeni i fiksni, nego da isti na neki način vrše adaptaciju realnom stanju na saobraćajnicama.

Sprovođenje bilo kojeg od ovih metoda je neophodno, kako bi saobraćanje nastavilo da se odvija zadovoljavajuće, te kako bi emisija štetnih plinova bila koliko god je moguće manja.

1.2 Sažetak predloženog rješenja, ciljevi rada i metodologija

Usljed svega pomenutog, u ovom radu će se uzeti za primjer glavna saobraćajnica u glavnom gradu Bosne i Hercegovine, Sarajevu, te jedna od najopterećenijih raskrsnica na njoj. Budući da je rekonstrukcija glavne saobraćajnice, uz pomoć npr. kružnih tokova, gotovo nemoguća, radi više infrastrukturnih i ekonomskih prepreka, jedino rješenje koja se ukazuje jeste adaptivno upravljanje svjetlosnom saobraćajnom signalizacijom koja i trenutno upravlja ovom saobraćajnicom.

Svrha svega onoga što će biti opisano unutar ovog rada jeste da se jedna raskrsnica na glavnoj sarajevskoj saobraćajnici učini optimalnom, uz jednaku infrastrukturu koja postoji od ranije. Dakle, jedino što se želi promijeniti jeste način smjene svjetlosnih saobraćajnih signala.

Za opisani zadatak koristit će se algoritam jedne od nekoliko glavnih oblasti mašinskog učenja: pojačano učenje (reinforcement learning). Osnovna ideja koja se sprovodi u opisanom metodu jeste da se definira "agent" koji će dobiti zadatak da, u različitim situacijama, na osnovu svog znanja, pokušava da ostvari najbolje moguće vrijednosti u datom trenutku koje opisuju zagušenost/protočnost raskrsnice, uz datu mogućnost izmjene trenutnog stanja u kojem se nalazi raskrsnica. Svaki put kada donese odluku, agent će analizirati kako je data odluka uticala na saobraćaj, te će na osnovu toga ažurirati svoje znanje, koje će vremenom konvergirati u znanje sa kojim je moguće optimalno upravljanje raskrsnicom, neovisno o tome u kojem se stanju raskrsnica nalazi u tom momentu.

1.3 Sažetak rezultata

Na način kako je to prikazano, između ostalih, i u radovima [1, 2, 3], ovaj rad, naposljetku, pokazuje kako adaptacija saobraćajne signalizacije trenutnom stanju na saobraćajnici, kao što je i očekivano, može, čak i uz vrlo jednostavnu implementaciju, značajno povećati performanse saobraćanja.

Ovo je potkrijepljeno korištenjem metrika, koje su, pored radova koji se bave implementacijom sličnih rješenja, naznačene i opisane i u literaturi koja se bavi izričito pitanjima saobraćaja, terminologije, metoda i tehnika vezanih za pomenutu tematiku (kao npr. [4]). Osnovne i najčešće korištene su opisane i u ovom radu u Poglavlju 2.2

Detaljni prikaz poboljšanja performansi saobraćanja korištenjem rješenja predloženog ovim radom, nalazi se u poglavlju 4.1

1.4 Struktura rada

Ovaj rad je sačinjen od 4 poglavlja, od kojih je u, trenutnom, Poglavlju 1, opisan problem koji se razmatra, motivacija rješavanja istog, sažetak predloženog rješenja koje će biti detaljno opisano u radu, ciljevi rada i metodologija.

Poglavlje 2 se referira na najznačajnije radove literature koja se bavi ovom oblašću, uz naglasak na načine definiranja problema koji se rješava, povijesni pregled pristupa i metoda korištenih za rješavanje istog, te metrike korištene za određivanje uspješnosti istih.

Poglavlje 3 opisuje metodološki okvir koji je primijenjen u ovom istraživanju. Detaljno se razmatra koncept pojačanog učenja (RL), uključujući njegove elemente i proces učenja, s posebnim naglaskom na Q-Learning algoritam. U nastavku se daje opis simulacionog okruženja, modela agenta i funkcije nagrade, kao i proces treniranja agenta. Na kraju poglavlja navode se ograničenja i pretpostavke korištenog modela.

Poglavlje 4 prikazuje rezultate eksperimentalnog dijela rada i daje diskusiju o postignutim rezultatima. Analizira se uspješnost predloženog rješenja u odnosu na saobraćajnu signalizaciju sa fiksnim ciklusima, i navode određene potencijalne mogućnosti za poboljšanje sistema.

Poglavlje 5 sumira postignuća rada, govoreći o načinu na koji je urađena implementacija na visokom nivou, te naglašava šta su naredni koraci u trenutnom stadiju razvoja sistema, akcentirajući mogućnost sigurne i pouzdane primjene u realnom svijetu.

Na samom kraju rada, u odjeljku Literatura, navode se svi korišteni izvori koji su poslužili kao teorijska i tehnička osnova za izradu ovog rada.

Poglavlje 2

Pregled literature/ Pregled postojećih sistema

Prvi korak ka boljem razumijevanju problema jeste da se predstavi na koji način se problem definira u literaturi.

Na temu adaptivnog upravljanja saobraćajem je napisan veliki broj radova, od kojih su neki orijentirani isključivo prema opisu problema koji se rješava, te načinu mjerenja uspješnosti nekog rješenja, ili s druge strane radovi koji su usmjereni isključivo ka opisu, vrstama i načinima implementacije pravih adaptivnih algoritama.

Za razliku od navedenih, oni koji su najznačajniji za ovaj rad, i koji su pružili najviše informacija, jesu radovi koji kombiniraju znanja obje oblasti kako bi pružili detaljan uvid u način implementacije adaptivnog upravljanja saobraćajnom signalizacijom.

U nastavku poglavlja će biti opisan problem koji se obrađuje u ovom radu, na način kako je to urađeno u najznačajnijim djelima napisanim u pomenutoj oblasti. To podrazumijeva opis i analizu najrelevantnijih radova iz ove teme te osvrt na sljedeće stavke:

1. Kako se u literaturi definira problem?
2. Kratka historija načina na koji se problem rješavao - tj. koje metode su se koristile kroz historiju, koje su njihove prednosti i mane, za koji (pod)problem (u generalnom problemu) su iste pogodne. Na koji način se metode porede (evaluacija).
3. Na koju skupinu metoda se fokusira ovaj rad, i zašto.

2.1 Definicija problema u literaturi

Prvenstveno, potrebno je napraviti opću definiciju problema koji se rješava, dakle, na koji način se on definira u pomenutoj literaturi. Ukoliko se napravi zbirni opis na osnovu [1, 2, 3, 5], problem adaptivnog upravljanja saobraćajnom signalizacijom se može formalno opisati kao optimizacija vremenskog rasporeda (faza i trajanja) semafora u saobraćajnoj mreži, s ciljem minimiziranja negativnih utjecaja prometa (što predstavljaju između ostalog: zagušenje, kašnjenja, emisije štetnih plinova, otežan transport resursa...), uzimajući u obzir dinamičku i stohastičku prirodu saobraćajnih tokova, koja je objašnjena u [1].

Nekoliko ključnih pojmova ovakve definicije, zahtijevaju dodatno objašnjenje:

1. Saobraćajna mreža - koncept koji je u svim radovima (implicitno ili eksplicitno) definiran kao sistem raskrižja i saobraćajnica, gdje svako raskrižje ima jedan ili više semafora.

2. Saobraćajni tok - izazov opisan u [1] kao kretanje vozila kroz saobraćajnu mrežu, koje se mijenja tokom vremena

3. Semafori su uređaji koji reguliraju promet na način dodjeljivanja prava prolaza različitim saobraćajnim strujama u različitim vremenskim fazama. Trajanje i odabir ovih faza je dio ključne diskusije u radovima kao [1, 5], kao i u predstojećim poglavljima ovog rada, budući da su predložena rješenja iz ovih radova zasnovana na pojačanom učenju, gdje se ograničeno upravljanje semaforima prepušta tzv. agentima koji pokušavaju da optimizuju saobraćajni tok

4. Minimizacija negativnih utjecaja saobraćaja - motiv koji je implicitno opisan u svim radovima, i predstavlja težnju ka uklanjanju ili smanjenju neželjenih popratnih pojava u saobraćajnoj mreži.

2.2 Metrike efikasnosti upravljanja saobraćajnom signalizacijom

Da bi se utjecaji saobraćaja bolje opisali, potrebno je definisati jasnu metriku koja će ocijeniti performans/učinkovitost neke metode upravljanja saobraćajnom signalizacijom. Cilj je postići određeni ekstrem (maksimum ili minimum) neke od tih metrika, da bi se minimizirali negativni utjecaji saobraćaja. Metrike koje se koriste i spominju u literaturi su:

1. Broj vozila, na osnovu [3], predstavlja ukupni broj vozila u saobraćajnoj mreži u određenom vremenu. Računa se kao razlika broja vozila koja dolaze i odlaze iz mreže u zadatom vremenskom periodu. Može dati preciznu indikaciju nivoa zagušenja mreže

2. Ukupno srednje kašnjenje - T_{AD} se koristi prema [3], dok je "n" broj raskrsnica a T_D je kašnjenje uzrokovano pojedinačnom raskrsnicom i T_N je broj vozila koja su puštena u mrežu.
$$T_{AD} = \sum_{i=1}^{n} \frac{T_{D_i}}{T_N} \tag{2.1}$$

3. Dužine redova čekanja - prema [4] jedna od dvije osnovne metrike korištene za izolovane raskrsnice. Redovi čekanja se mogu određivati na razne načine za različite potrebe, ali najintuitivniji način jeste računanje dužine saobraćajnice koja je zauzeta vozilima koja čekaju da dobiju signal za dozvolu kretanja. Vrlo je bitno za ovaj rad naglasiti da se dužine redova čekanja računaju po svakoj traci koja prilazi raskrsnici, budući da mnoge raskrsnice, kao i ona korištena u ovom radu, nemaju jednak broj traka, pa samim tim ni jednak kapacitet protoka iz svih smjerova.

4. Kašnjenje uzrokovano signalizacijom - u [4] definirano kao druga od dvije osnovne metrike korištene za izolovane raskrsnice, a predstavlja dio ukupnog kašnjenja učesnika u saobraćaju, koji je uzrokovan bilo kojom vrstom signalizacije. Ukupno kašnjenje, sa druge strane, jeste ukupna razlika vremena potrebnog da učesnik prođe kroz saobraćajnu mrežu sa i bez: drugih učesnika, signalizacije, nepredviđenih okolnosti itd.

Pored pomenutih, u literaturi je moguće pronaći dodatne metrike kojima se mjeri performans upravljanja saobraćajem. Međutim, veliki broj tih metrika su veoma teško mjerljive, kao npr. metrika "nivoa izduvnih gasova uzrokovanih saobraćajnom signalizacijom", budući da ovakve metrike podrazumijevaju veliku količinu instrumentacije i obrade podataka.

Budući da je izbor metrike koja se koristi za mjerenje performansi algoritma koji upravlja signalizacijom, usko povezan sa konkretnim algoritmom koji se koristi, u ovom radu će prvo biti predstavljen kratki povijesni razvoj pomenutih algoritama, uključujući izazove na koje su isti naišli, nakon čega će biti predstavljen algoritam koji je obrađen u ovom radu, kao i metrika koja određuje učinak ovog algoritma u odnosu na saobraćajnu signalizaciju sa fiksnim rasporedom i trajanjem faza.

2.3 Povijesni pregled i poređenje algoritama iz oblasti

Kako se to navodi u [1], važno je napomenuti da svi algoritmi za rad u stvarnom vremenu nisu nužno adaptivni. Ta dva pojma se često koriste naizmjenično i mogu se poistovijetiti, unatoč značajnoj razlici između istih.

Algoritmi za rad u stvarnom vremenu su oni koji mogu reagovati na senzorske ulaze u stvarnom vremenu, iako im interna logika i parametri kontrolera ostaju nepromijenjeni. S druge strane, bitna karakteristika adaptivnih algoritama je njihova sposobnost da prilagode svoju internu logiku i parametre kao odgovor na značajne promjene u okruženju — promjene koje mogu učiniti bazu znanja u neadaptivnom kontroleru zastarjelom. Obzirom na prethodno navedeno, ovaj rad će se bazirati na pojačano učenje, i jedan konkretan algoritam iz ove oblasti mašinskog učenja, budući da isti, kako navodi [1] jesu stvarno adaptivni, u smislu da su sposobni, ne samo imati real-time epitet, nego i taj da dinamički mijenjaju svoje djelovanje, putem kontinuiranog učenja i prilagođavanja.

Uprkos tome, ovdje će biti navedeni i radovi koji obrađuju i druge načine rješavanja ovog problema, zarad predstavljanja toka (ne)uspjeha da se saobraćajna signalizacija učini "pametnom".

Historijski, upravljanje semaforima započinje fiksno vremenskim planovima ("fiksni ritam") u kojima su dužine faza unaprijed određene i ne reaguju na promjene. Ovakvi sistemi su jednostavni, ali neefikasni pri promjenjivim uslovima jer ne koriste informacije o stvarnoj gustoći saobraćaja. Standardni priručnici, npr. [4], ističu da adaptivni sistemi u saobraćaju mogu odložiti zasićenje raskrsnice i brže se oporaviti nakon zagušenja, dok fiksni planovi ponekad bolje odgovaraju izuzetno nisko protočnim raskrižjima.

2.3.1 Prvi koraci

Krajem 20. stoljeća uvedeni su aktuirani sistemi koji mijenjaju trajanje svjetlosne signalizacije na osnovu detektora prisutnosti vozila. Ove metode su često bile usmjerene na pojedinačne raskrsnice ili lokalne grupe raskrsnica uz minimalnu koordinaciju, te su značajno poboljšale performanse u odnosu na fiksne planove u umjereno dinamičkim uslovima.

2.3.2 Fuzzy logika

Sa razvojem računarskih mreža i naprednih senzora pojavili su se sistemi adaptivnog upravljanja poput SCATS-a i SCOOT-a 1980-ih, spomenuti u [1, 2, 3], gdje se svjetlosni planovi stalno prilagođavaju realnom vremenu na osnovu mjerenja protoka i zagušenja. U ovoj fazi pojavili su se i heuristički i ekspertni pristupi, kao što su *fuzzy* logika i genetski algoritmi, također navedeni u [1, 2, 3]. *Fuzzy* logika omogućava modeliranje neizvjesnosti i ljudskih pravila: npr. "ako je saobraćaj gust na sjever-jugu, produži zeleno" umjesto fiksnih pragova. Ovakve metode često su temeljene na stručnom iskustvu i specifičnoj konfiguraciji raskrsnice. Iako su pokazale poboljšanja, one zahtijevaju ručnu kalibraciju pravila i ne uče autonomno iz okruženja.

2.3.3 Početak primjene RL algoritama

Kako se povećavala moć računarstva i razvoja umjetne inteligencije, krajem 20. stoljeća istraživači su primijetili da učenje potkrepljenjem (**R**einforcement **L**earning) može autonomno poboljšavati kontrolu semafora bez detaljnog modeliranja saobraćaja. U [1] demonstrirana je primjena jednostavnog Q-učenja (koji predstavlja RL algoritam) na upravljanje jednim pravougaonim raskrižjem dvosmjernih saobraćajnica, pokazujući da agent, samostalno učeći relaciju između akcija i efektata u saobraćaju, može ostvariti značajno smanjenje prosječnog kašnjenja vozila u odnosu na pretpostavljeni sistem fiksnih intervala. Ovaj rad je otvorio put širem istraživanju RL pristupa: umjesto fiksnih modela okoline, agenti koriste iterativnu optimizaciju kroz interakciju sa stvarnim ili simuliranim protokom saobraćaja. U [5] se naglašava da je RL "autonomni" pristup u adaptivnom upravljanju signalizacijom, važan za rješavanje problema zagušenja u pametnim gradovima.

2.3.4 Multi-agentska RL rješenja

Daljnji razvoj uključivao je multi-agentske RL sisteme gdje svaki semafor ili grupa semafora ima svog agenta. Na primjer, u [3] je predstavljena distribuirana arhitektura u kojoj agenti na susjednim raskrsnicama dijele podatke o protoku i mrežnim uslovima te koriste RL da prilagode dužinu zelene svjetlosti u cilju minimiziranja ukupnog kašnjenja. U studijama simulacija u Singapuru ovaj multi-agentski RL sistem ostvario je značajno poboljšanje prosječnog kašnjenja i brzine kretanja vozila u odnosu na konkurentske kontrolere (hierarhijski sistem, kooperativni ansambl i obični aktuirani kontroler). Time je pokazano da RL agenti mogu efikasno koordinisati lokalne odluke na nivou cijele mreže, premda je i ovdje izazov ne-stacionarna priroda prometa (u toku učenja) i potreba za bržom konvergencijom.

2.3.5 Deep RL algoritmi

U posljednjem desetljeću zabilježen je napredak dubokog pojačanog učenja (Deep RL). Integracija dubokih neuronskih mreža omogućava agentima da direktno obrađuju sirove ili složene ulazne podatke (poput slika sa kamera ili velikog niza senzorskih podataka) i da uče reprezentacije bez ručnog inženjeringa karakteristika. U [2] je navedeno da je sa pojavom dubokog učenja potkrepljenjem sa neuronskim mrežama, taj pristup pokazao bolje performanse i robusnost u odnosu na tradicionalne RL metode, zahvaljujući boljoj generalizaciji u visoko-dimenzionalnim stanjima. U [5] je dalje istaknuto da ova duboka integracija omogućava agentima da izgrađuju sofisticirane strategije (npr. korištenjem metoda poput DQN (Deep Q-Network), DDPG (Deep Deterministic Policy Gradient), attention RL i drugih) za globalno optimiziranje protoka saobraćaja. Ipak, duboki RL nosi i svoje izazove – veći zahtjevi za podacima i računarskom snagom,

te složenije treniranje – pa je često kombinovan s tradicionalnim metodama (npr. inicijalizacija s klasičnim kontrolerom) kako bi se ubrzao proces učenja.

2.3.6 Savremeni pristupi

Ukupno gledajući, savremeni pristupi za adaptivno upravljanje semaforima sve više koriste RL: od klasika poput Q-učenja iz [1], preko višelagentskih okvira kao u [3], do najnovijih deep RL arhitektura kao što su one opisane u [2, 5]. Ovi algoritmi, kako je to dokazano i naglašeno radovima [2, 3, 5] omogućavaju kontrolerima da samostalno uče i prilagođavaju se promjenama u saobraćaju, čak i u velikim mrežama, za razliku od klasičnih metoda koje se oslanjaju na fiksne modele ili empirijske pravilnike.

U nastavku je data tabela usporedbe ključnih pristupa i algoritama u adaptivnom upravljanju signalizacijom, uključenih u historijski razvoj i savremene istraživačke trendove.

Metoda	**Godina**	**Tip metode**	**Prednosti**	**Mane**
Fiksno-vremenski kontroler	1950-1960	Fiksni / offline	Jednostavan hardver	Veliko kašnjenje u zagušenju
Aktuirani kontroler	1970-e	Dinamički / offline	Brža reakcija na promjene	Ograničena koordinacija i dodatni detektori
Fuzzy-logika kontroler	1980-e	Heuristički / inteligentni	Dobar kod stohastičkih ulaza	Nije autonoman i zahtijeva tuning pravila
Q-learning	2003	RL / Jednoagentski	Jednostavna implementacija s minimalnim pretpostavkama	Teško skaliranje na velike mreže
Multi-agentni kontroler	2010	RL / Distribuirani	Povećana skalabilnost i koordinacija raskrsnica	Zahtijeva sinhronizaciju i složenije učenje
Deep Q-Network	2016	Duboki RL / Neuronske mreže	Dobra generalizacija i performanse radi dubine	Zahtjevan trening osjetljiv na hiperparametre
Optimizacija u mreži	2017	Duboki i multi-agentni RL	Potencijal za globalno optimiziranje	Potreba za puno zahtjevnih simulacija

Tabela 2.1: Poređenje algoritama iz oblasti

2.3.7 Sažetak

U konačnici, tradicionalni pristupi pružaju jednostavno rješenje, ali loše performanse u dinamičkim uvjetima. Moderni algoritmi na pojačanom učenju, pogotovo u višelagentskim i

dubokim oblicima, omogućavaju samostalno prilagođavanje signalizacije stvarnom vremenu protoka, čime se u teoriji može postići značajno bolje upravljanje zagušenjem i protočnosti u usporedbi sa starijim metodama. Međutim, takvi sistemi istovremeno zahtijevaju veću računarsku snagu, dodatnu hardversku opremu i pažljivu kalibraciju kako bi bili praktični za primjenu u stvarnim mrežama saobraćaja.

2.4 Specifikacija opsega rada

Ovaj rad je fokusiran na implementaciju **jednoagentskog** rješenja za adaptivno upravljanje semaforima, zasnovanog na algoritmu **Q-learning**, sličnog kakav je opisan u [1]. Za razliku od višelagentskih i distribuiranih pristupa, ovdje jedan agent donosi odluke o fazama i trajanju svjetlosnih signala na osnovu stanja saobraćaja, koje se diskretizira prema unaprijed definisanim metrikama.

Eksperimentalno okruženje ovog rada obuhvata **jednu odabranu raskrsnicu u Sarajevu**, koja je modelirana u simulacionom alatu SUMO na temelju stvarnih podataka o strukturi saobraćajnice. Ovakav fokus omogućava detaljnu analizu performansi Q-learning agenta u kontrolisanom, ali realističnom scenariju, te poređenje s tradicionalnim fiksno-vremenskim planom za istu lokaciju.

Poglavlje 3

Metodološki pristup

U ovom poglavlju bit će predstavljeni

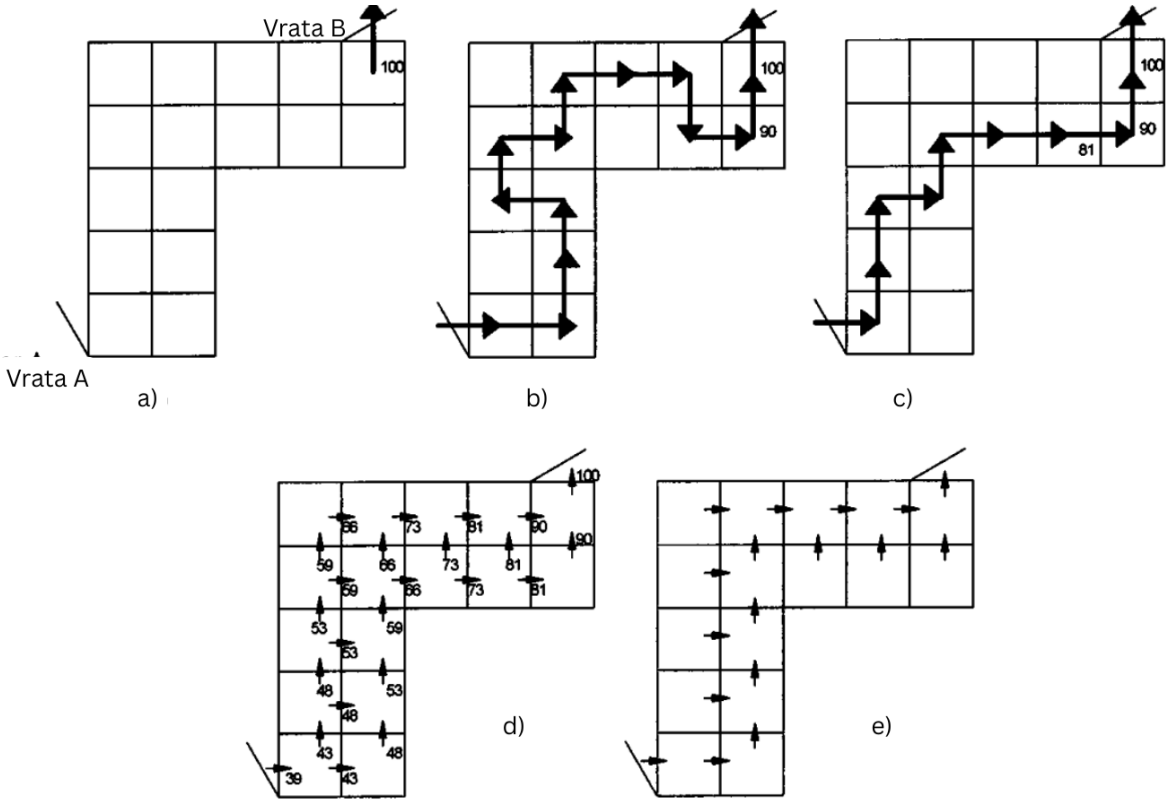
1. Definicija problema koji se rješava, odnosno postavka hipoteza koja jasno proizilazi iz uvoda i pregleda literature.
2. Opis pristupa koji se koristi za adresiranje postavljenog problema / hipoteze, što podrazumijeva:
   (a) Opis podataka: način na koji su podaci predstavljeni kao ulazni parametar za simuliranje saobraćajne raskrsnice,
   (b) Metode pripreme podataka za analizu,
   (c) Koji algoritmi su korišteni, njihova postavka (postavka parametara, hiperparametara itd.), sa kojim ciljem, tj. zašto su korišteni baš ti algoritmi,
   (d) Mjere performansi koje su korištene za usporedbu algoritama (zašto baš te mjere).

3.1 Pojačano učenje/*Reinforcement Learning*

U najjednostavnijim crtama, pojačano učenje podrazumijeva agenta koji želi naučiti kako da ostvari neki cilj. To postiže tako što dinamički komunicira sa svojom okolinom, isprobava različite akcije u različitim situacijama kako bi utvrdio najbolju ili niz najboljih za postizanje cilja iz bilo koje početne situacije. Signali povratnih informacija iz okoline omogućavaju agentu da odredi u kojoj mjeri je određena akcija doprinijela ostvarenju željenog cilja.

Da ilustrujemo koncept pojačanog učenja, razmotrimo pojednostavljen primjer mobilnog robota koji se kreće u mrežastom prostoru prikazanom na slici 1(a). Ovo je zapravo ilustracija Q-*learninga*, koji je razvio Watkins (1989; Watkins i Dayan 1992), kako je naglašeno u radu [1], te predstavlja samo jedan od mogućih algoritama pojačanog učenja, te je upravo on korišten u rješenju predstavljenom kasnije u radu.

Zamislimo da robot počinje iza vrata A i da mu je cilj proći kroz vrata B, za što dobija nagradu od maksimalno 100 jedinica. Druge akcije se ne nagrađuju. Kada prođe kroz vrata B, ostaje tamo (možda čekajući novi zadatak) i ne dobija nove nagrade. U svakom vremenskom koraku robot se može pomjeriti u susjedno polje mreže, ali ne može se kretati dijagonalno. Dakle, robot je u svakom trenutku svjestan samo u kojim pravcima od mogućih četiri može da se kreće, odnosno s kojih strana postoji polje na koje može da se pomjeri, te je svjestan



Slika 3.1: Ilustracija učenja potkrepljenjem: (a) mrežni svijet; (b) prva epizoda; (c) druga epizoda; (d) odabrane konačne Q-procjene; (e) jedna moguća optimalna polisa. [1]

ukoliko se nalazi pored vrata B, koja su njegov cilj. Definišemo i stopu diskontovanja koja smanjuje vrijednost budućih nagrada u odnosu na one koje se dobijaju odmah. Pod ovim se misli na umanjenje nagrade koja može doći zbog akcije *a* u istoj epizodi/pokušaju, ali tek nakon određenog broja pokušaja, ne odmah nakon izvršenja akcije *a*. U ovom slučaju, diskontna stopa dodatno motiviše robota da nauči najkraći mogući put do vrata B. Pretpostavimo da je diskontna stopa 0.9.

Na početku, sve moguće akcije iz bilo kojeg polja, osim one koja vodi direktno kroz vrata B iz polja ispred njih, imaju vrijednost nula, jer se čini da njima ne može biti ostvarena nagrada. Na svom prvom putovanju robot se, dakle, kreće nasumično, možda prateći put prikazan na slici 1(b), sve dok ne prođe kroz vrata B. Tada dobija nagradu od 100 jedinica i ostaje na tom mjestu, čime završava trenutna epizoda. Vrijednost akcije koja je prethodila ulasku kroz vrata B se dodjeljuje ili ažurira na osnovu nagrade od 100 jedinica, pomnožene sa diskontnom stopom 0.9 (jer je nagrada dobijena jedan korak unaprijed), što daje neto vrijednost od 90 jedinica.

Na drugom putovanju od vrata A, robot se ponovo kreće dok ne stigne do polja pored onog ispred vrata B. Kao i prije, akcija koja prethodi posljednjoj ažuriranoj (ili dodijeljenoj) akciji dobija vrijednost jednaku vrijednosti posljednje ažurirane akcije pomnožene diskontnom stopom 0.9, što u konkretnom primjeru, na slici 1(c), predstavlja 90 jedinica pomnoženo diskontnom stopom 0.9, što daje 81 jedinicu. Svaka naredna epizoda može dovesti do toga da još neka akcija dobije vrijednost. U nekom trenutku, robot se može naći pred izborom između akcije koja ima vrijednost 0 i one koja ima prethodno dodijeljenu, pozitivnu vrijednost. Tada se mora odlučiti

da li će istražiti novu opciju (koja trenutno ima vrijednost 0) u nadi da je bolja, ili će iskoristiti ono što već zna i odabrati poznatu akciju s pozitivnom vrijednošću.

Nakon dovoljno epizoda, svaka akcija iz bilo kojeg polja dobit će određenu vrijednost. U većini praktičnih problema, posebno u stohastičkim okruženjima, potrebno je mnogo epizoda da bi se ove vrijednosti stabilizovale. Slika 1(d) prikazuje dio tih vrijednosti, gdje svaka predstavlja zbir diskontovanih budućih nagrada ako se slijedi optimalni put od tog polja do vrata B. Robot je, dakle, naučio procjenu Q-funkcije. Q-funkcija je funkcija koja za dato stanje i akciju procjenjuje očekivanu kumulativnu nagradu koju će agent dobiti ako nastavi da postupa najbolje moguće u datom trenutku, te, kada je naučena, agent može primijeniti optimalnu polisu tako što uvijek bira akciju s najvećom vrijednošću, bez obzira odakle kreće, sve dok ne stigne do vrata B (slika 1(e)).

3.1.1 Elementi pojačanog učenja

Pojačano učenje sastoji se od nekoliko ključnih elemenata koji zajedno definišu način interakcije agenta sa okolinom, a sve navedene stavke su grafički predstavljene i povezane na Slici 3.2

Agent

Agent je entitet koji donosi odluke na osnovu informacija koje dobija iz okoline. Njegov cilj je da, tokom interakcije, nauči polisu odlučivanja (*policy*) koja maksimizira očekivanu kumulativnu nagradu. U kontekstu upravljanja saobraćajem, agent predstavlja kontroler semafora koji bira da li će se faza signalizacije na raskrsnici promijeniti ili ostati u trenutnoj.

Okolina

Okolina predstavlja sve ono sa čim agent vrši interakciju i od čega dobija povratne informacije. Ona se mijenja kao rezultat akcija agenta i vanjskih faktora. U problemu adaptivnog upravljanja saobraćajem, okolina je saobraćajna mreža (u ovom slučaju simulirana u SUMO simulatoru), uključujući vozila, raskrsnice, trake, pravila kretanja i mnoge druge elemente.

Stanje

Stanje (*state*) je reprezentacija trenutne situacije u kojoj se okruženje nalazi. Ono mora sadržavati dovoljno informacija da agent može donijeti informisanu odluku. Primjeri stanja kod saobraćajnog semafora uključuju broj vozila po traci, prosječno vrijeme čekanja ili zagušenost na prilazima raskrsnici.

Akcija

Akcija (*action*) je odluka ili potez koji agent izvršava u određenom stanju. U slučaju adaptivnog semafora, akcija može biti promjena trenutne faze signalizacije, produžavanje zelene faze ili prelazak na sljedeću fazu.

Nagrada

Nagrada (*reward*) je povratna informacija koju agent dobija od okoline nakon izvršene akcije. Ona kvantitativno opisuje kvalitet odluke u datom stanju. Kod upravljanja saobraćajem, nagrada se često definiše kao negativna vrijednost ukupnog vremena čekanja, dužine kolona ili zagušenja, kako bi agent bio motivisan da ih minimizira, kao što je predstavljeno u radovima [1, 3].

3.1.2 Petlja pojačanog učenja

Proces pojačanog učenja odvija se iterativno kroz niz koraka koji zajedno čine tzv. *petlju pojačanog učenja*. Ova petlja se ponavlja tokom cijelog procesa treniranja agenta, sve dok ne postigne zadovoljavajuće performanse ili dok se ne ispuni unaprijed definisan kriterij zaustavljanja. Glavni koraci su sljedeći:

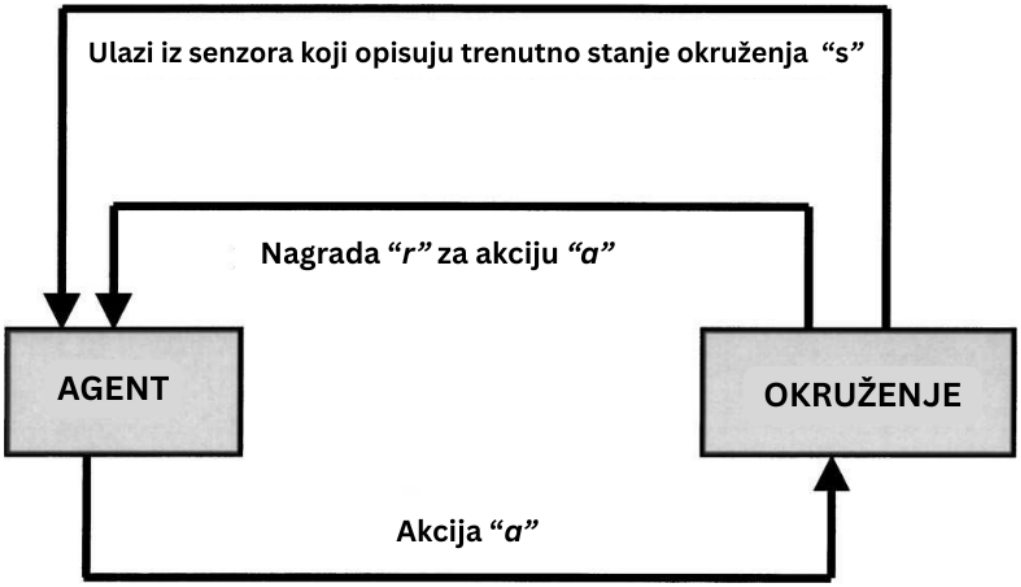
1. **Posmatranje stanja okoline:** U početku svake iteracije, agent opaža trenutno stanje s_t iz okoline.
2. **Odabir akcije:** Na osnovu trenutnog stanja i polise odlučivanja (npr. ε-*greedy*), agent bira akciju a_t koju će izvršiti.
3. **Izvršenje akcije i prijelaz u novo stanje:** Akcija a_t se primjenjuje na okolinu, što rezultira prelaskom u novo stanje s_{t+1}.
4. **Primanje nagrade:** Agent dobija nagradu r_t koja odražava kvalitet odabrane akcije u odnosu na definisani cilj.
5. **Ažuriranje polise:** Koristeći dobijenu nagradu i novo stanje, agent ažurira svoju procjenu korisnosti akcije u datom stanju, što će biti izvedeno u ovom slučaju pomoću Q-*learning* pravila.
6. **Ponavljanje:** Petlja se nastavlja prelaskom na novo stanje s_{t+1} i ponovnim izvršavanjem prethodnih koraka, sve dok epizoda ne završi.

Ovakva struktura omogućava agentu da kroz kontinuiranu interakciju s okolinom poboljšava svoje odluke, maksimizirajući kumulativnu nagradu tokom vremena.

3.1.3 Precizna definicija Q-*learning* algoritma

Na osnovu prethodnih pojašnjenja, sada će biti predstavljena preciznija, iako i dalje osnovna, definicija Q-*learning*a, jedne vrste RL algoritma, koja će biti korištena u sklopu predstavljenog rješenja u ovom radu.

Ova definicija je u sličnom obliku data u [1] budući da se okvirni princip implementacije rješenja u ovom radu također u velikoj mjeri zasniva na rješenju predstavljenom u pomenutom referenciranom djelu.



Slika 3.2: Ključni elementi Q-*learning* algoritma [1]

Sve navedene stavke su grafički predstavljene na Slici 3.2

- Agent je entitet odgovoran za tumačenje ulaznih podataka iz okoline, biranje akcija na osnovu tih podataka i učenje iz posljedica svojih akcija na okolinu. U trenutku t, Q-*learning* agent od okoline dobija signal koji opisuje trenutno stanje s. Teoretski, informacija o stanju mora imati Markovljevo svojstvo, što znači da je dovoljno poznavati trenutno stanje i izabranu akciju da bi se predvidio rezultat u sljedećem stanju — nije potrebno znati historiju svih prethodnih stanja ili akcija. U praksi se obično pretpostavlja da proces jeste Markovljev, iako to ne mora biti potpuno tačno.

- Na osnovu svog opažanja stanja s, agent bira akciju a iz skupa mogućih akcija. Odluka zavisi od relativne vrijednosti različitih mogućih akcija, tj. procijenjenih Q-vrijednosti $Q(s,a)$, koje predstavljaju vrijednost poduzimanja akcije a u stanju s, nakon čega slijedi prelazak u novo stanje s' i nastavak po trenutno optimalnoj polisi. Na početku, agent nema nikakve procjene Q-vrijednosti i mora ih naučiti istražujući nasumične akcije. Postepeno prelazi sa istraživanja (*exploration*) na iskorištavanje (*exploitation*) onih kombinacija stanja i akcija koje su se pokazale dobrima.

- Kao rezultat preduzimanja akcije a u stanju s, agent dobija nagradu $r(s,a)$, koja zavisi od uticaja te akcije na okolinu. Može postojati kašnjenje između izvršene akcije i primljene nagrade. Cilj agenta je pronaći optimalnu polisu koja maksimizira ukupnu nagradu (ili minimizira ukupnu kaznu) tokom vremena. Diskontna stopa može se koristiti da ograniči nagradu, posebno u slučajevima kontinuiranih epizoda.

- Kombinacija stanja s, akcije a i nagrade $r_{s,a}$ se zatim koristi za ažuriranje prethodne procjene Q-vrijednosti $Q_{t-1(s,a)}$ rekurzivno prema sljedećem pravilu treniranja:

$$\delta = \alpha_{s,a}\left\{r_{s,a} + \gamma_t \cdot \max\left[Q_{t-1(s',a')}\right] - Q_{t-1(s,a)}\right\} \tag{3.1}$$

gdje je δ = inkrement koji se dodaje prethodno procijenjenoj Q-vrijednosti $Q_{t-1(s,a)}$ da bi se dobila $Q_{t(s,a)}$; $\alpha_{s,a}$ = stopa učenja u intervalu $[0,1]$; $r_{s,a}$ = nagrada dobijena za preduzimanje akcije a dok je agent u stanju s; γ_t = stopa diskontiranja u intervalu $[0,1]$, primijenjena na buduće nagrade; $\max\left[Q_{t-1(s',a')}\right]$ = prethodno procijenjena Q-vrijednost prateći optimalnu polisu počevši od stanja s'; i $Q_{t-1(s,a)}$ = prethodna procjena Q-vrijednosti za preduzimanje akcije a dok je agent u stanju s.

Ovo pravilo je posebno korisno u stohastičkim okruženjima, poput saobraćajnog sistema, kao što je naglašeno u [1]. Za konvergenciju Q-funkcije u takvom okruženju, potrebno je da se stopa učenja smanjuje tokom vremena i da svaka kombinacija stanja i akcije bude posjećena beskonačno mnogo puta (u praksi, najvažniji dio prostora stanja biće posjećen vrlo često, iako ne beskonačno). Ako se umjesto nagrada dobijaju kazne, umjesto MAX koristi se MIN funkcija.

- Ažurirana procjena Q-vrijednosti se pohranjuje za kasniju upotrebu. Q-vrijednosti se mogu čuvati u obliku tabele, iako to može zahtijevati dosta memorije. Također, mogu se koristiti u aproksimaciji funkcija kako bi se procijenile Q-vrijednosti za stanja i akcije koje agent još nije posjetio, ali su slične onima koje jeste.

3.2 Opis simulacionog okruženja

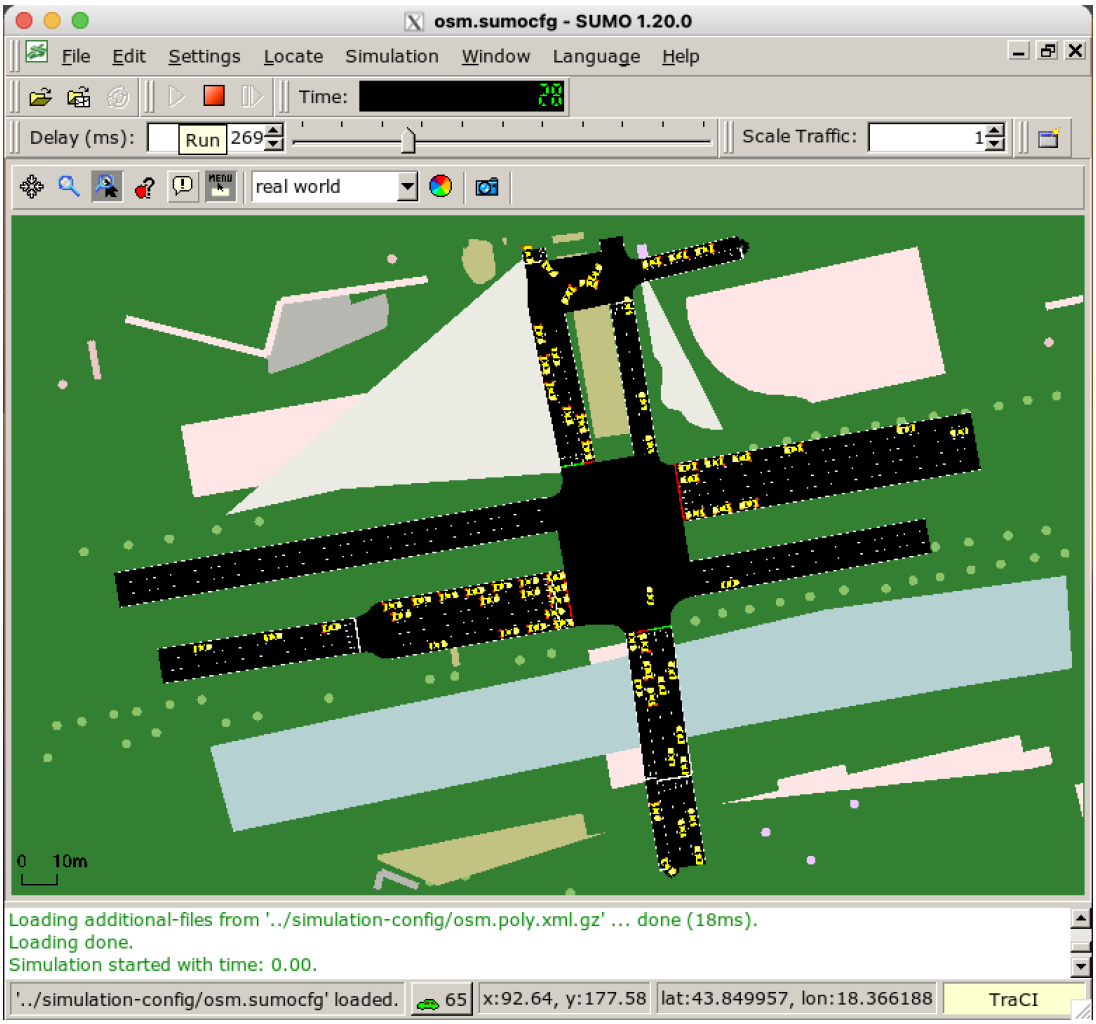
Simulaciono okruženje koje je korišteno za modeliranje i analizu saobraćajnih tokova je SUMO (*Simulation of Urban MObility*) simulator, koji predstavlja moćan i fleksibilan alat za mikrosimulaciju saobraćaja.

Za interakciju i kontrolu simulacije korišten je TraCI (Traffic Control Interface) API, koji omogućava upravljanje simulacionim elementima u realnom vremenu putem Python koda. Korištenjem TraCI-ja, moguće je dinamički mijenjati stanje, kako semafora tako i mnogih drugih elemenata raskrižja i toka saobraćaja, prikupljati podatke o vozilima i analizirati saobraćajne parametre tokom trajanja simulacije.

Ispod, na slici 3.3 prikazan je snimak ekrana simulacije koji ilustruju izgled i tok simulacionog procesa.

SUMO simulator pruža mogućnost odabira okvira sa realne mape svijeta koja uključuje registrirane saobraćajnice, što, dakako, ostavlja mogućnost da se ovaj alat koristi i za obimnije simulacije koje obuhvataju veću urbanu zonu sa mnoštvom raskrsnica, tačno onakvih kakve jesu u realnom svijetu, a ne samo one izolovane, ili ručno modelirane.

Važno je napomenuti da se u SUMO simulatoru konfiguracija nekog dijela saobraćajne mreže može eksportovati u obliku foldera koji sadrži skup `.xml` fajlova. Ovi fajlovi detaljno opisuju različite aspekte saobraćajnih parametara, poput definicije puteva, raskrsnica, semaforskih planova i ruta vozila. Konkretno, raskrsnica koja je predmet istraživanja u ovom radu rekonstruisana je upravo iz takvog seta fajlova, a cjelokupna konfiguracija, zajedno sa svim programskim kodom korištenim za implementaciju i testiranje algoritama, dostupna je u repozitoriju [6].



Slika 3.3: SUMO simulator

3.3 Model agenta

U okviru ovog rada razvijen je Q-*learning* agent primjenjiv za adaptivno upravljanje semaforima na raskrsnici bilo kakve formacije. Njegova struktura i ponašanje nisu predstavljeni uopšteno, već su precizno prilagođeni domenskom problemu saobraćaja. Agent se zasniva na metodi pojačanog učenja bez modela (eng. *model-free reinforcement learning*) i uči optimalnu polisu kroz direktnu interakciju sa SUMO simulacijom, postupnim ažuriranjem Q-vrijednosti.

Za implementaciju su definisane sljedeće ključne funkcionalnosti:

- **Diskretizacija stanja putem `get_state_key`**. Budući da su sirova stanja iz simulacije kontinuirana i vrlo visoke dimenzionalnosti (npr. precizan broj vozila u redu čekanja ili trajanje trenutne faze semafora), agent bi bez dodatne obrade imao enormno velik prostor stanja, što bi onemogućilo efikasno učenje. Da bi se taj problem riješio, uvedena je funkcija `get_state_key` koja *mapira slična stanja u iste diskretne binove*.
  - Broj vozila u redu po traci ograničen je na maksimalno 60 što je gornja granica očekivanog obima vozila za simulacije koje su izvođene u toku treniranja i evaluacije agenta, a zatim grupisan u intervale veličine 5 (npr. red od 0 do 4 vozila se tretira kao isto stanje, 5 do 9 kao drugo, itd.).
  - Trajanje trenutne faze semafora dijeli se u binove od 10 sekundi, čime se smanjuje osjetljivost na minimalne promjene.

  Ovim pristupom se postiže značajno smanjenje prostora stanja jer se više realnih situacija koje su "dovoljno slične" tretiraju jednako, a time se ubrzava i stabilizuje proces učenja.

- **Polisa izbora akcije**. Agent bira akciju korištenjem ε-greedy pristupa, tj. sa vjerovatnoćom ε bira nasumičnu akciju (istraživanje), a sa vjerovatnoćom $1-\varepsilon$ bira akciju sa najvećom procijenjenom Q-vrijednošću (iskorištavanje). Na taj način osigurano je da agent ne zaglavi prerano u lokalnom optimumu, nego da nastavi istraživati potencijalno bolje strategije. U kontekstu ovog problema, agent raspolaže binarnim skupom akcija: *ostavi trenutnu fazu semafora* ili *promijeni fazu*.

- **Q-tabela i funkcija učenja**. Za svako diskretizirano stanje i akciju održava se Q-vrijednost. Ažuriranje Q-tabele vrši se standardnom Q-*learning* formulom. Radi bolje čitljivosti, formula se ovdje predstavlja u obliku:

$$Q_{\text{novo}}(s,a) \;=\; Q(s,a) \;+\; \alpha \left[r \;+\; \gamma \max_{a'} Q(s',a') \;-\; Q(s,a) \right] \tag{3.2}$$

  gdje:
  - $Q(s,a)$ predstavlja trenutnu procjenu kvaliteta akcije a u stanju s,
  - r je nagrada koju agent dobija od simulacije nakon izvršene akcije,
  - α je stopa učenja kojom agent kontroliše koliko se oslanja na novodobijenu informaciju,
  - γ je faktor diskontovanja koji smanjuje značaj budućih nagrada u odnosu na trenutne,
  - s' je novo stanje u koje sistem prelazi nakon akcije a,

– $\max_{a'} Q(s', a')$ je procjena vrijednosti najbolje moguće akcije u novom stanju s'.

Ova formula osigurava da se agent ne oslanja samo na neposredne nagrade, već i na dugoročne posljedice svojih odluka.

Detaljan pseudokod Q-*learning* agenta prikazan je u Pseudokodu 1, a kompletna Python implementacija se nalazi u Prilogu 2.

Algoritam 1: Pseudokod za QLearningAgent

Podaci: Skup akcija A, parametri α (learning rate), γ (discount factor), ε (exploration rate)

Rezultat: Naučena Q-tabela za odabir optimalnih akcija

Inicijalizacija:
$q\_table \leftarrow \emptyset$;
postavi $\alpha, \gamma, \varepsilon, A$;

Funkcija `GET_STATE_KEY(state)`**:**
  podijeli *state* na $(phase, duration, queues)$;
  diskretizuj redove čekanja u intervale veličine $QUEUE\_STEP$;
  diskretizuj *duration* u binove od 10 sekundi ;
  return $(phase, duration\_bin, queue\_bins)$;

Funkcija `GET_Q(state, action)`**:**
  $key \leftarrow (GET\_STATE\_KEY(state), action)$;
  if $key \in q\_table$ **then**
    return $q\_table[key]$;
  else
    return 0.0 ;

Funkcija `CHOOSE_ACTION(state)`**:**
  if $random() < \varepsilon$ **then**
    $a \leftarrow$ nasumično iz A ;
  else
    izračunaj $Q(state, a)$ za sve $a \in A$;
    $Q\_max \leftarrow \max Q(state, a)$;
    $a \leftarrow$ nasumično iz skupa $\{a : Q(state, a) = Q\_max\}$;
  return a ;

Funkcija `LEARN(state, action, reward, next_state)`**:**
  $current\_q \leftarrow GET\_Q(state, action)$;
  $next\_max\_q \leftarrow \max\_a \in AGET\_Q(next\_state, a)$;
  $new\_q \leftarrow current\_q + \alpha \cdot (reward + \gamma \cdot next\_max\_q - current\_q)$;
  $q\_table[(GET\_STATE\_KEY(state), action)] \leftarrow new\_q$;

3.4 Okruženje i predstavljanje stanja

Okruženje u kojem agent uči i djeluje modelirano je, kako je već navedeno, u SUMO simulatoru, pri čemu se adaptivno upravljanje semaforima ostvaruje preko TraCI interfejsa. U ovom kontekstu, okruženje obuhvata sve elemente raskrsnice koji su potencijalno relevantni za donošenje odluka: faze semafora, trajanje trenutne faze, dužine redova vozila na svakom prilazu, prosječno čekanje vozila, kao i dinamiku promjene saobraćajnog toka tokom vremena. Konkretno, u pogledu pojačanog učenja, okruženje predstavlja sve ono što je moguće opservirati, prema tome, u ovom slučaju okruženje predstavljaju sve vrijednosti i parametri koje je moguće dobiti kroz TraCI.

Međutim, nije svaki element jednako koristan niti praktičan za uključivanje u reprezentaciju stanja. Potrebno je napraviti kompromis između informativnosti stanja i složenosti problema. Ako se u stanje uključi previše varijabli, dimenzionalnost problema značajno raste, što otežava ili čak onemogućava proces učenja i konvergenciju Q-*learning* algoritma ka optimalnoj polisi. Sa druge strane, previše uprošteno stanje dovodi do gubitka informacija i slabijih performansi agenta.

Prvobitni pokušaji uključivali su praćenje dužine redova iz svih pravaca, ali je to dovelo do problema jer različiti prilazi imaju različit broj traka. To je uzrokovalo neuravnotežene podatke i otežalo formiranje konzistentnog stanja. Također, eksperimentisano je sa proširenom reprezentacijom koja je uključivala prosječno vrijeme čekanja po traci. Iako ova mjera pruža dodatnu informaciju o kvalitetu saobraćajnog toka, njenim uključivanjem drastično se povećala dimenzionalnost problema, što je dovelo do sporog ili nikakvog učenja. Iz tih razloga, ova reprezentacija je napuštena.

Konačna verzija reprezentacije stanja fokusira se na sljedeće elemente:

- trenutna faza semafora - korisna informacija koja agentu pomaže da razluči akcije prema fazama semafora u kojima se nalazi, jer ista akcija, sa istim redovima čekanja, nije uvijek najbolja u svakoj fazi semafora, nego zavisi od toga u kojoj se fazi semafor trenutno nalazi. Ova informacija je posebno značajna kada semafor ima veći broj faza, kao u ovom slučaju gdje semafor ima osam faza,
- trajanje trenutne faze - informacija koja olakšava agentu da raspozna koliko trajanje treba dati svakoj od faza semafora kako bi replicirao akcije iz prošlosti koje su dale dobre rezultate,
- prosječan broj vozila po traci za svaki prilaz (umjesto ukupnog broja vozila, čime se normalizuje razlika između prilaza sa različitim brojem traka) - informacija koja čini glavni dio stanja koji će se koristiti kako bi agent mogao biti objektivno nagrađen/kažnjen, odnosno kako bi njegove akcije bile ispravno evaluirane.

Ova kombinacija pokazala se kao balans između informativnosti i upravljivosti, omogućavajući agentu da donosi odluke na osnovu dovoljno podataka, ali bez eksplozije dimenzionalnosti.

Također, bitno je naglasiti da se broj vozila po traci iz svakog smjera, prije vraćanja iz funkcije, sortira, tako da će respektivne vrijednosti u zapisu stanja uvijek predstavljati količinu vozila za isti smjer kretanja.

Sljedeći pseudokod prikazuje implementaciju funkcije koja vraća trenutno stanje okruženja na osnovu definisanog semafora:

Algoritam 2: Pseudokod funkcije `get_state` za predstavljanje stanja okruženja

Podaci: Identifikator semafora tls_id
Rezultat: Triplet koji opisuje trenutno stanje $(faza, trajanje, redovi)$

$trenutna\_faza \leftarrow getPhase(tls\_id)$;
$trajanje\_faze \leftarrow getPhaseDuration(tls\_id)$;
$prilazi \leftarrow \emptyset$;
$traka\_ukupno \leftarrow \emptyset$;
foreach $traka$ *u* $getControlledLanes(tls\_id)$ **do**
    $smjer\_id \leftarrow traka.smjer$;
    if $smjer\_id \notin prilazi$ **then**
        $prilazi[smjer\_id] \leftarrow 0$;
        $traka\_ukupno[smjer\_id] \leftarrow 0$;
    $prilazi[smjer\_id] \leftarrow prilazi[smjer\_id] + getLastStepVehicleNumber(traka)$;
    $traka\_ukupno[smjer\_id] \leftarrow traka\_ukupno[smjer\_id] + 1$;
foreach $smjer\_id$ *u* $prilazi$ **do**
    $prilazi[smjer\_id] \leftarrow prilazi[smjer\_id] / traka\_ukupno[smjer\_id]$;
$sorted\_prilazi \leftarrow sort(prilazi)$;
$duzine\_redova \leftarrow [q \mid (edge, q) \in sorted\_prilazi]$;
return $(trenutna\_faza, trajanje\_faze) + duzine\_redova$;

3.5 Funkcija nagrade

Funkcija nagrade predstavlja jedan od najbitnijih elemenata Q-*learning* algoritma. Način na koji je ona definisana uveliko određuje šta agent pokušava postići i kakva ponašanja će tokom procesa učenja razvijati. U suštini, agent uči kroz interakciju s okruženjem tako što dobija nagrade ili kazne na osnovu posljedica svojih akcija. Stoga, kvalitetno dizajnirana funkcija nagrade može značajno ubrzati proces učenja, dok neadekvatna definicija može potpuno onemogućiti konvergenciju algoritma.

U ovom radu funkcija nagrade koncipirana je tako da penalizuje dva aspekta saobraćajnog toka:

1. **Prosječne redove čekanja po traci** – gdje se kvadratno kažnjava dužina reda. Na ovaj način se naglašava razlika između kratkih i dugačkih redova, jer dugi redovi imaju nesrazmjerno veći negativni uticaj na saobraćaj.
2. **Preduge faze semafora u uslovima gužve** – ukoliko se dogodi da faza traje predugo, a pritom su redovi čekanja veliki, uvodi se dodatna kazna proporcionalna vremenu trajanja faze preko određenog praga.

Konačna implementacija funkcije nagrade prikazana je kroz sljedeći pseudokod, izveden na osnovu Python koda korištenog u eksperimentima:

Algoritam 3: Pseudokod funkcije nagrade

Podaci: Stanje $s = (faza, trajanje, redovi)$
Rezultat: Skalarna vrijednost nagrade r

$faza, trajanje, duzine\_redova \leftarrow s$;
$kazna\_red \leftarrow \frac{1}{|redovi|} \cdot \sum_{q \in redovi} q^2$
if $\max(duzine\_redova) > 20 \ \wedge \ trajanje > 60$ **then**
  $kazna\_trajanje \leftarrow (trajanje - 60) \cdot 1.2$;
else
  $kazna\_trajanje \leftarrow 0$;
$nagrada \leftarrow -(kazna\_red + kazna\_trajanje)$;

Bitno je napomenuti da su postojali pokušaji definisanja funkcije nagrade s bogatijim informacijama, poput prosječnog vremena čekanja vozila ili detaljnijeg praćenja protoka saobraćaja. Ipak, uvođenje takvih dodatnih parametara znatno povećava dimenzionalnost problema. Kao posljedica toga, agent teže "razumije" uzročno-posljedičnu vezu između svojih akcija i dobijene nagrade. Ovaj fenomen dovodi do toga da proces učenja postaje znatno sporiji, pa čak i nemoguć, jer agent ne može efikasno generalizovati u velikom prostoru stanja.

Iz tog razloga, odabrana je jednostavnija varijanta funkcije nagrade koja koristi samo kvadratnu penalizaciju dužine reda i dodatnu kaznu za preduge faze u uslovima gužve. Na ovaj način se postigao balans između informativnosti nagrade i njene praktične upotrebljivosti u kontekstu učenja Q-*learning* agenta.

3.6 Proces treniranja

Trening Q-*learning* agenta realizovan je kroz ukupno **2200 epizoda**. Svaka epizoda predstavlja jednu nezavisnu simulaciju u SUMO okruženju u toku koje agent komunicira sa simulacijom putem TraCI interfejsa, bira akcije, prikuplja nagrade i ažurira Q-tablicu.

Postavke i hiperparametri

Za treniranje su korišteni sljedeći početni hiperparametri i stope prigušenja:

- Početna stopa učenja: `ALPHA = 0.187`
- Faktor diskontovanja: `GAMMA = 0.95`
- Početna eksploracija: `EPSILON = 1.0`
- `ALPHA_DECAY = 0.9996` (u svakoj epizodi: `alpha *= ALPHA_DECAY`)
- `EPSILON_DECAY = 0.997` (u svakoj epizodi: `epsilon *= EPSILON_DECAY`)

Za potrebe randomizacije ulaznog saobraćaja korišteni su seedovi iz skupa $[0, 6]$. Svaka epizoda započinje generisanjem slučajnog saobraćaja pomoću funkcije za generisanje ruta (SUMO `randomTrips`), pri čemu su vozila generisana između `800` i `1100` sekundi, uz protok od otprilike **2500–3500 vozila/h**. Drugim riječima, prije pokretanja glavne simulacijske petlje za svaku epizodu generiše se saobraćaj po prethodno definisanim pravilima i seed vrijednosti.

Struktura jedne epizode

Sama epizoda implementirana je u funkciji `run_episode`, koja je priložena u prilogu Program 3, a u kojoj se, ukratko, izvršavaju sljedeće akcije:

1. Pokretanje SUMO simulacije pomoću `traci.start(...)` sa konfiguracijom iz `CONFIG_FILE`.
2. Inicijalizacija metrika: broja pokretanja i dolazaka vozila, ukupne nagrade, kumulativno čekanje i sl.
3. Inicijalno očitavanje stanja
4. Glavna simulacijska petlja:
    - Prate se brojevi otpremljenih i pristiglih vozila radi određivanja trenutka završetka simulacije.
    - Prikupljaju se vremena čekanja svih vozila radi kasnije evaluacije.
    - Određuje se trenutna faza semafora i, ako su zadovoljeni vremenski uvjeti (`step - last_action_time >= MIN_PHASE_DURATION`), bira se akcija:
        - Ako je proteklo više od `MAX_PHASE_DURATION`, prisilno se odabire akcija koja mijenja fazu.
        - Inače se koristi agentova polisa

- Nakon izvršenja eventualne promjene faze, izračunava se nagrada te se akumulira u `total_reward`.
- Agent uči pozivom `agent.learn(current_state, action, reward, next_state)`.
- Ažuriraju se kumulativne statistike (departures/arrivals).

5. Nakon izlaska iz petlje simulacija se zatvara (`traci.close()`) i računa se prosječno čekanje kao `avg_waiting` = `cumulative_waiting/measurement_count`.
6. Funkcija vraća uređenu n-torku: $(total\_reward, step, sim\_generating\_end, arrived\_vehicles, avg\_waiting)$, koji se koristi za logovanje i evaluaciju.

Logovanje, checkpointi i evaluacija

Praćenje napretka i checkpointiranje izvedeno je kako slijedi:

- Nakon svake epizode zapisuje se red u log fajl `q-tables-and-logs/log.csv` u formatu: `Episode,Total Reward,Gen End,Sim End,Arrived Vehicles,Avg Waiting`.
- Q-tabela se serijalizuje i čuva u direktorijumu `q-tables-and-logs/tables/` na svakoj 40. epizodi i na posljednjoj epizodi.
- Po događaju čuvanja Q-tabele pokreće se skripta za evaluaciju: `evaluate_agent.py`, što omogućava periodične provjere performansi naučene politike bez ručnog intervenisanja.
- Nakon svake epizode ažurira se konfiguracijski fajl pozivom funkcije `update_config(...)` kako bi se u fajlu pohranile posljednje vrijednosti hiperparametara (`last_alpha`, `last_gamma`, `last_epsilon`) i broj dovršenih epizoda (`episodes_done`).

Napomene o stabilnosti učenja i dizajnu

U dizajnu procesa treniranja posebno se vodilo računa o stabilnosti i reproduktivnosti:

- Randomizacija početnog saobraćaja (seedovi 0–6) smanjuje *overfitting* na specifične šablone protoka saobraćaja i omogućava učenje robusnijih pravila.
- Postepeno smanjivanje ε i α pomaže u tranziciji od istraživanja ka stabilnom učenju i smanjuje oscilacije u polisi.
- Čuvanje check-pointova i periodična evaluacija omogućavaju uvid u trendove performansi i vraćanje na ranije točke u slučaju potrebe.

Ovaj postupak treniranja daje jasan, reproducibilan i mjerljiv tok eksperimenta: prije svake epizode generiše se (seed-ovan) saobraćaj, u toku epizode agent donosi odluke i uči iz nagrada, a rezultati i stanje učenja se redovno loguju i snimaju radi daljnje analize i evaluacije.

3.7 Ograničenja i pretpostavke modela

Ograničenja modela

- **Simulacijski jaz (simulation-to-reality gap):** SUMO model i simulirane vožnje ne hvataju sve nijanse stvarnog saobraćaja (npr. neočekivana ponašanja vozača, kvarovi vozila, vremenski uslovi). To može dovesti do razlika između performansi u simulaciji i u realnom okruženju. Međutim, očekivano je da se veliki dio poboljšanja u simulaciji, jednako odrazi i u stvarnom okruženju.
- **Diskretizacija stanja:** Stanje je diskretizovano (kvadratna suma broja vozila, prosječno čekanje itd.), što pojednostavljuje problem ali ograničava sposobnost modela da reprezentuje finije varijacije u prometu.
- **Ograničen skup senzora/observation space:** Model pretpostavlja dostupnost i točnost metričkih podataka koje SUMO vraća; u realnom sistemu senzori mogu biti ograničeni, bučni ili nedostupni.
- **Jedan agent / lokalna kontrola:** Implementacija cilja za pojedinačni semafor (ili nekoordinirane lokalne kontrolere). Nije obuhvaćena koordinacija više raskrsnica kroz multi-agentnu strategiju, što može ograničiti performanse u mrežnom prometu.
- **Statičnost pravila vozila i ruta:** Model koristi unaprijed definisane profile vozila i generator ruta (seed-ovi 0–6). Promjene u vrstama vozila ili ponašanju (npr. udio autobusa, dostavnih vozila) nisu eksplicitno modelirane.
- **Skalabilnost i dimenzionalnost:** Klasični Q-*learning* s tabličnom Q-tabelom ne skalira dobro s velikim brojem diskretnih stanja/akcija (curse of dimensionality). Za veće, realističnije prostore stanja bit će potrebna aproksimacija funkcije (npr. DQN).
- **Ovisnost o obliku funkcije nagrade:** Izbor i oblik nagrade snažno utječu na naučenu polisu; suboptimalno definirana nagrada može dovesti do neželjenog ponašanja (npr. favoriziranje samo jedne trake).
- **Resursi i vrijeme treninga:** Potrebno je značajno računarsko vrijeme za hiljade epizoda i multiple seed-ove; to ograničava brzinu iteracija i eksperimentisanja s hiperparametrima.
- **Procjena performansi:** Evaluacija u radu se oslanja na metrike iz simulacije (prosječno čekanje, ostvaren broj dolazaka). Te metrike možda neće obuhvatiti sve aspekte kvalitete (npr. varijansu čekanja među vozilima, utjecaj na pješački promet).

Pretpostavke modela

- **Pouzdanost simulacijskih podataka:** Pretpostavlja se da SUMO generiše konzistentne i statistički reprezentativne scenarije saobraćaja unutar definisanih parametara (800–1100 s trajanja generacije; 2500–3500 vozila/h protok).
- **Homogenost vozila:** U eksperimentima se pretpostavlja da tipovi vozila i njihovo ponašanje ne variraju značajno (osnovni model vozača). Specifične varijante (npr. električni automobili s različitim ubrzanjem) nisu posebno tretirane.

- **Potpuna i tačna opažanja u svakom koraku:** Agent prima potrebne vrijednosti stanja u svakom vremenskom koraku bez kašnjenja ili gubitka podataka.
- **Fiksna struktura semaforskih faza i akcija:** Skup dozvoljenih akcija i struktura faza su fiksni i unaprijed definirani (npr. promjena faze ili održavanje iste); agent ne mijenja dizajn faza.
- **Stacionarnost unutar epizode:** Tijekom jedne epizode saobraćajni obrasci se smatraju dovoljno stacionarnim za svrhu učenja; drastične, iznenadne promjene potražnje unutar epizode nisu modelirane.
- **Ovisnost o odabiru seed-ova:** Randomizacija se svodi na seed-ove 0–6; pretpostavlja se da ta pokrivenost daje reprezentativnu varijaciju prometnih scenarija za ciljnu domenu.
- **Epsiode ne utiču međusobno osim kroz checkpoint-ove:** Svaka epizoda je nezavisna simulacija; jedino trajno stanje je sačuvana Q-tabela i spremljeni hiperparametri.

Ideje/koraci za ublažavanje ograničenja

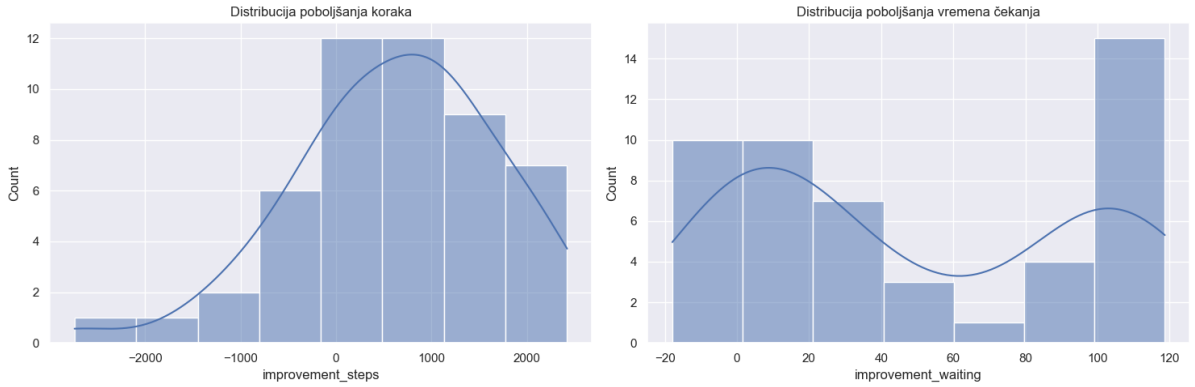
- Proširenje skupova seed-ova i scenarija (uključujući varijacije u udjelu tipova vozila, vremenskim uslovima i incidentima) radi smanjenja simulacijskog jaza.
- Prelazak na aproksimatore funkcije (neuralne mreže, DQN) za bolje skaliranje u većim prostorima stanja.
- Dodavanje mjera u nagradnu funkciju i uvođenje dodatnih metrika kako bi se izbjeglo ”lokalno najbolje” ponašanje.

Poglavlje 4

Evaluacija

Krajnja evaluacija performansi agneta je izvršena uz pomoć skripte *evaluate_agent.py*, ali ovaj put uz pokretanje 50 različitih epizoda po dva puta, pri čemu je u prvi put raskrsnica upravljana semaforom sa fiksnim intervalima, a drugi put je prepuštena istreniranom agentu da upravlja fazama semafora. Nakon dobijenih rezultata simulacija, sačuvanih u CSV fajlu, detaljna analiza dobijenih rezultata je urađena sa drugom skriptom koja je iskorištena da generiše najznačajnije grafove koji pokazuju značaj i unapređenje agentskog pristupa upravljanju saobraćajnom signalizacijom. Pomenuti grafovi i diskusija vezana za iste nalaze se ispod.

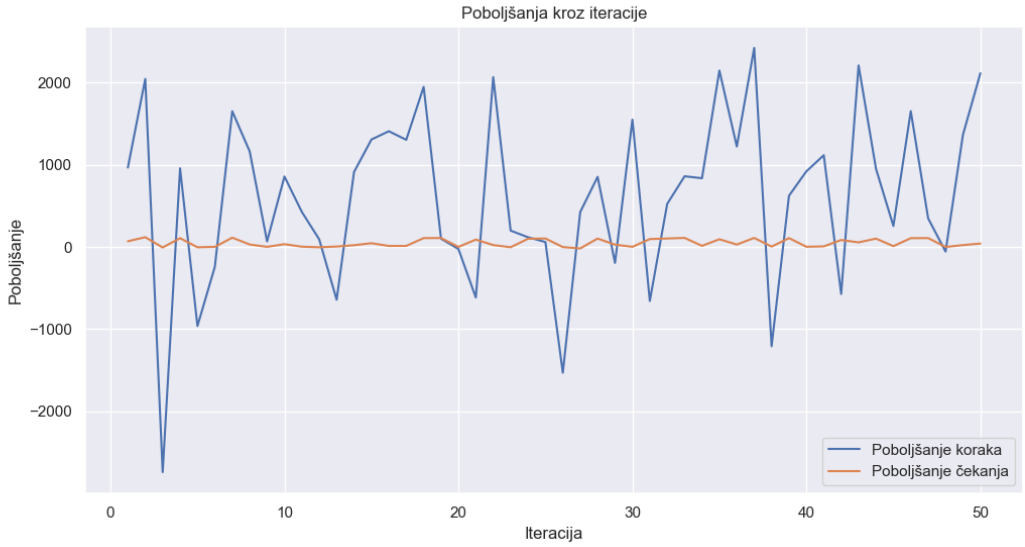
4.1 Diskusija rezultata



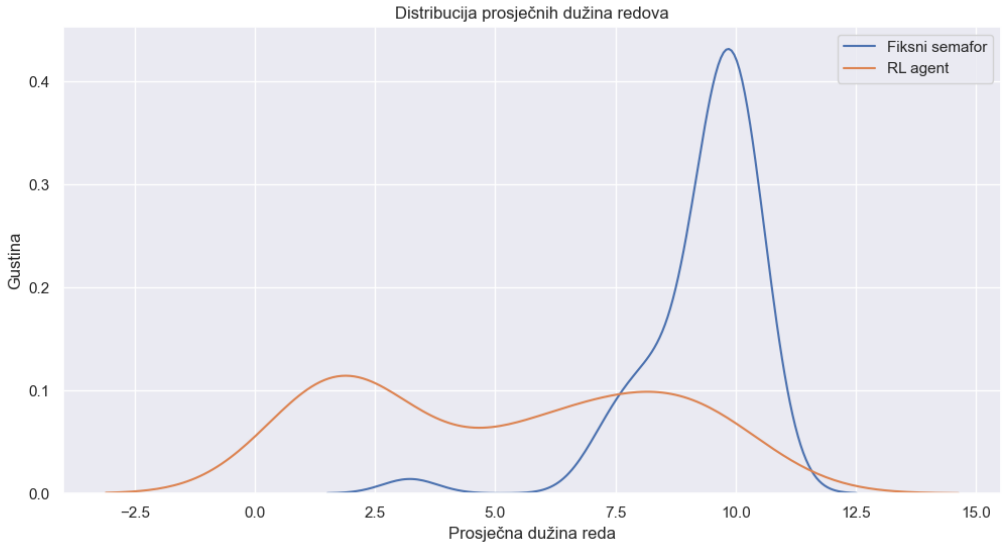
Slika 4.1: Distribucija poboljšanja (1) sekundi trajanja simulacije (2) sekundi čekanja po vozilu

U ovoj sekciji sumiramo i interpretiramo glavne nalaze vidljive na priloženim figurama (Slike 4.1–4.6) te izvodimo praktične zaključke i preporuke za dalji rad.

Opća opažanja o poboljšanjima Histogrami i KDE na Slici 4.1 (lijevo: poboljšanje broja koraka, desno: poboljšanje vremena čekanja) pokazuju da su poboljšanja raspoređena neravnomjerno — iako postoji vidljiv pomak u pozitivnom smjeru, prisutni su i negativni slučajevi. To sugerira da naučena polisa često dovodi do korisnog smanjenja čekanja, ali u određenim scenarijima (vjerojatno specifičnim seed-ovima ili visokim opterećenjima) agent može pogoršati performansu u odnosu na referentnu strategiju, što za sobom povlači činjenicu da usavršavanje opisa okruženja i definicije nagrade sigurno može napraviti još bolje rezultate.



Slika 4.2: Poboljšanje kroz iteracije

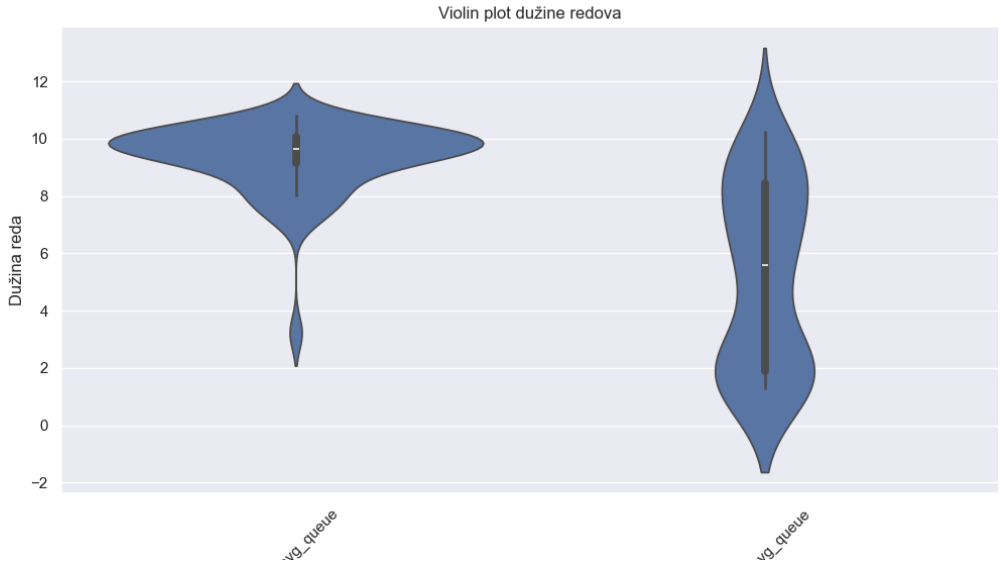


Slika 4.3: Distribucija redova čekanja

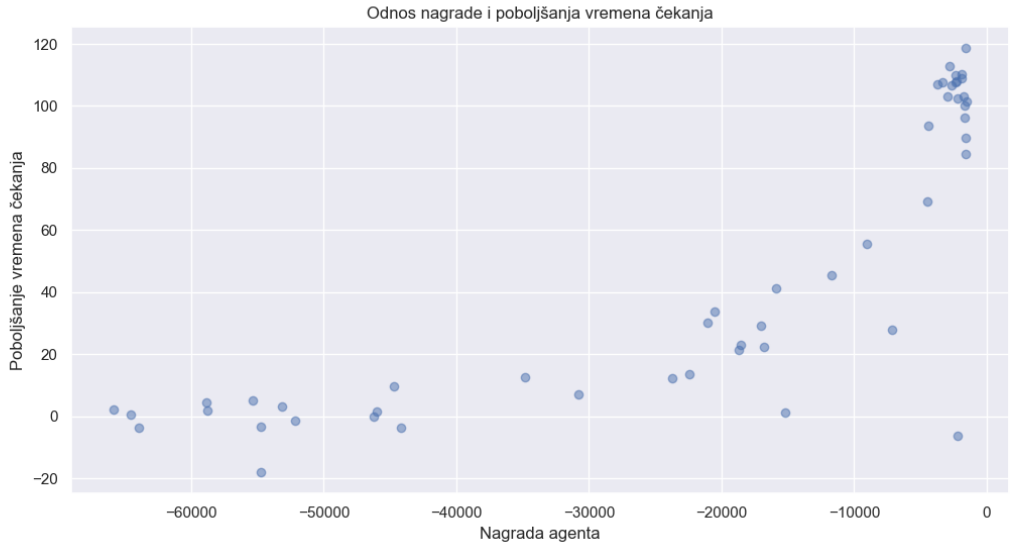
Redovi čekanja — distribucija i varijabilnost Slikom 4.3 i violinom na Slici 4.4 jasno se vidi razlika između fiksnog semafora i RL agenta:

- Fiksni semafor ima usku, oštru distribuciju s vrhom oko veće vrijednosti prosječne dužine reda (koncentracija oko 9–10), što znači dosljedno duga čekanja.
- RL agent pokazuje širu, često bimodalnu ili pomaknutu distribuciju: prosječna dužina reda je često manja, ali s većom varijansom i prisutnim vrlo kratkim redovima u nekim slučajevima.

Tumačenje: RL politika smanjuje srednju dužinu reda u većini scenarija, ali povećava varijansu — u nekim situacijama postiže znatno bolje rezultate nego fiksni semafor, dok u drugim može stvarati ekstremnije vrijednosti (iako u prosjeku povoljnije).



Slika 4.4: Violin graf redova čekanja (1) fiksni semafor (2) RL

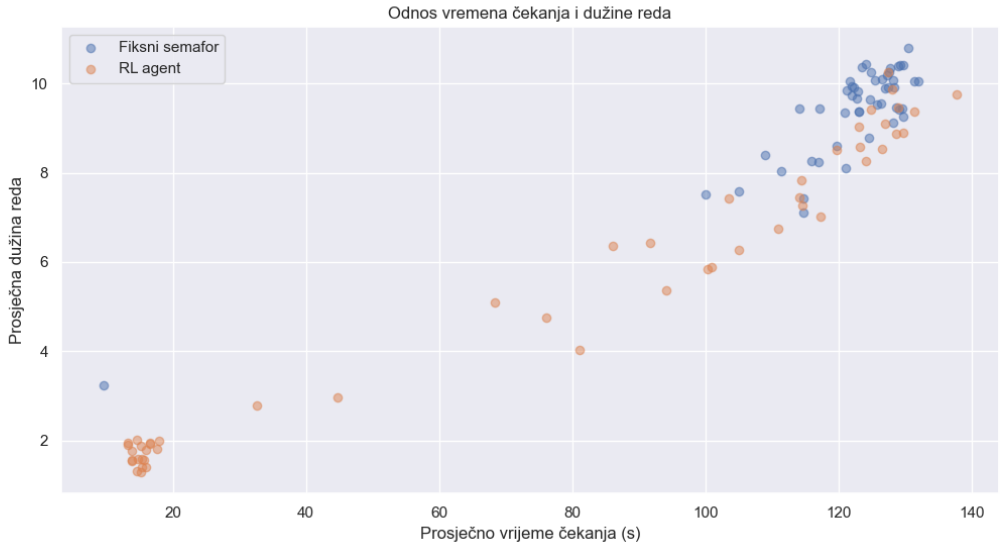


Slika 4.5: Poređenje nagrade i poboljšanja

Odnos nagrade i poboljšanja Raspršeni graf na Slici 4.5 pokazuje jasan trend: epizode s manjom (manje negativnom) ukupnom nagradom povezane su s većim poboljšanjem vremena čekanja. To potvrđuje opravdanost korištene nagradne funkcije (ona bar djelomično odražava smanjenje vremena čekanja), ali i pokazuje da apsolutne vrijednosti nagrade (koje su negativne i velike po apsolutnoj vrijednosti) mogu biti teške za interpretirati bez normalizacije. Evidentno, ovo predstavlja oblast u kojoj bi se standardizacijom i normalizacijom postigla još bolja tačnost i nagrada bi se još više približila stvarnom odrazu stanja u saobraćaju.

Distribucija vremena čekanja Slika 4.6 prikazuje kako fiksni semafor ima usko i visokobuketiranu distribuciju s jasnim vrhom na vrlo velikim vremenima čekanja (npr. oko 110s), dok RL agent ima razvučeniju distribuciju s nižim srednjim vrijednostima i dužim repovima.

Slika 4.6: Distribucija vremena čekanja



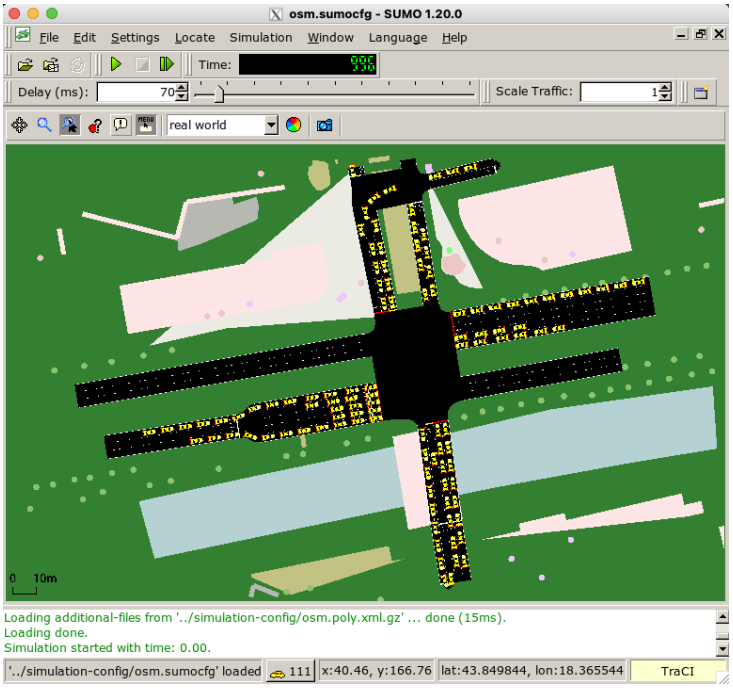
Slika 4.7: Poređenje vremena čekanja i dužine redova

Interpretacija:

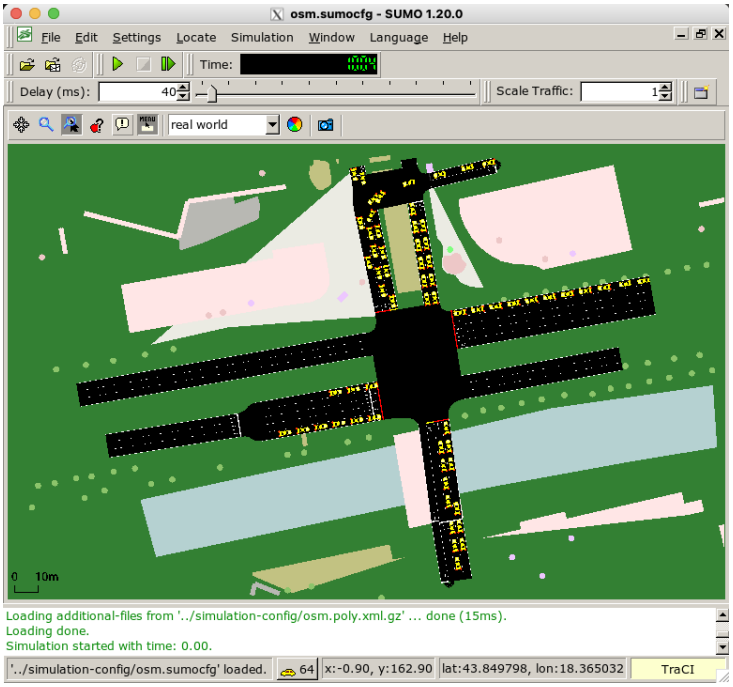
- RL smanjuje prosječno vrijeme čekanja u mnogim slučajevima (pomjeranje mase distribucije na niže),
- međutim, prisutan je i rep (rjeđe epizode s vrlo dugim čekanjima), što ukazuje na scenarije u kojima politika ne generalizira dovoljno dobro.

4.2 Stvarni odraz na stanje u saobraćaju

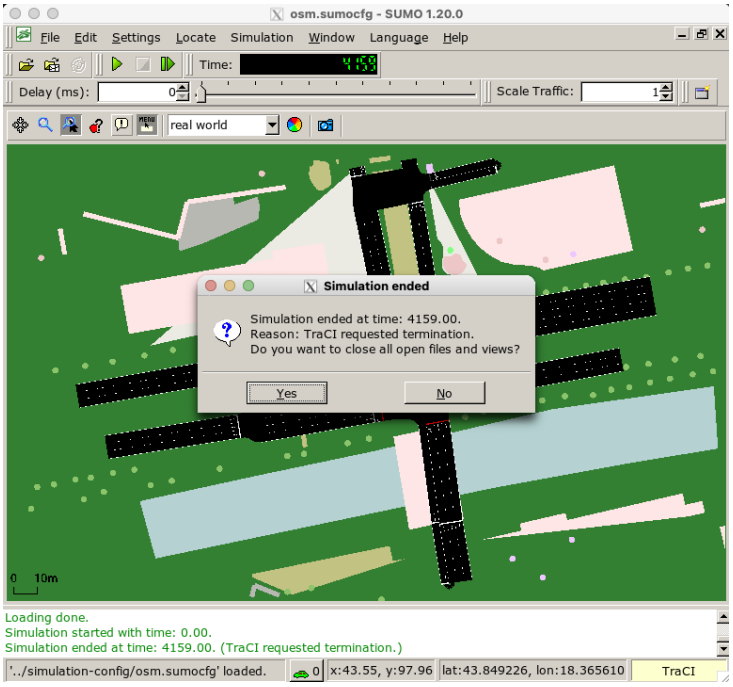
Na slikama ispod je jasno vidljivo da su prethodno predstavljene prednosti agentskog pristupa stvarne, tj. vidljive u nasumičnom trenutku u saobraćajnom toku.



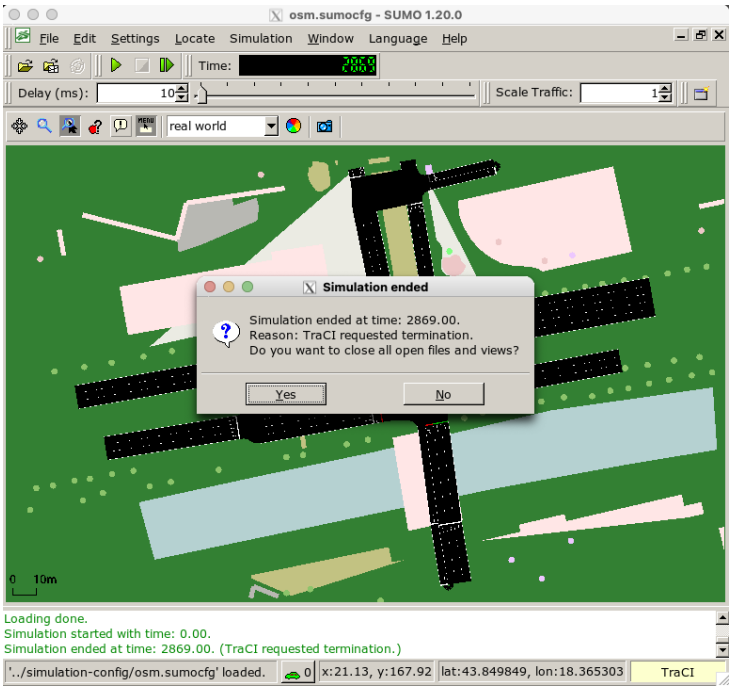
Slika 4.8: Simulacija sa semaforom s fiksnim intervalima (1000 sekundi od početka)



Slika 4.9: Simulacija sa semaforom upravljanim RL agentom (1000 sekundi od početka)



Slika 4.10: Kraj simulacije sa semaforom s fiksnim intervalima: 4159s



Slika 4.11: Kraj simulacije sa semaforom upravljanim agentom: 2869s

Poglavlje 5

Zaključak

U radu je pokazana primjena Q-learning agenta za adaptivnu kontrolu semafora u SUMO simulacijama. Cilj je bio provjeriti može li jednostavan, tablični Q-learning smanjiti prosječnu dužinu reda i vrijeme čekanja u odnosu na fiksnu, statičku kontrolu. Implementacija obuhvata trening na 2200 epizoda s periodičnim spremanjem Q-tablice, evaluacijom i logovanjem osnovnih performansi.

Glavni nalazi su sljedeći:

- RL agent u prosjeku smanjuje prosječnu dužinu reda približno 40% i prosječno vrijeme čekanja u odnosu na fiksni semafor, također približno 40%.
- Performanse agenta pokazuju određenu varijabilnost između epizoda, međutim potreba za vremenom pri izvršavanju treniranja je bila ograničavajući faktor za postizanje veće stabilnosti.
- Distribucije rezultata pokazuju da je dobit u prosjeku postignuta smanjenjem centralne mase (niža srednja vrijednost čekanja), ali uz povećanu varijansu.

Metodološki, tablični Q-learning se pokazao kao dobar početni izbor za dokaz koncepta zahvaljujući jednostavnosti implementacije i interpretabilnosti Q-tablice. Međutim, njegove limite (skalabilnost, osjetljivost na rijetka stanja i varijabilnost učenja) jasno ukazuju na potrebu za naprednijim pristupima prije eventualne primjene u realnom sistemu. Ovo je također objašnjeno u [1, 3, 5].

Kao zaključak: metoda demonstrira potencijal RL kontrole semafora — posebno u smanjenju prosječnih metrika — ali za praktičnu upotrebu potrebni su dodatni koraci radi poboljšanja stabilnosti, robusnosti i mogućnosti primjene na cijelu saobraćajnu mrežu.

5.1 Budući rad

Sljedeće aktivnosti predlažu se kao prioriteti za nastavak rada. Fokus je na smanjenju varijanse performansi, povećanju robusnosti i približavanju simulacijskih rezultata stvarnom okruženju.

Model i treniranje

- **Prelazak na aproksimatore funkcije (npr. DQN):** zamijeniti tablični Q-learning DQN-om ili sličnim metodama kako bi se bolje skaliralo s većim i kontinuiranim prostorom stanja i smanjila osjetljivost na rijetka stanja.

- **Eksperimenti s polisom i trening režimima:** testirati različite strategije EPSILON-decay, on-policy metode ili policy gradient pristupe za bolji balans istraživanja i eksploatacije.

Nagradna funkcija i evaluacija

- **Redefinisanje nagrade:** uvesti komponente koje penaliziraju visoku varijansu čekanja (npr. penal za 95. percentil) i/ili nagrade koje balansiraju throughput i pravičnost među trakama.
- **Dodatne metrike:** pri evaluaciji koristiti medijan, 75. i 95. percentil čekanja, varijansu, broj zaustavljenih vozila i ukupno vrijeme putovanja (travel time) — ne oslanjati se isključivo na prosjek.
- **Statistička validacija:** proširiti broj seed-ova i sprovesti statističke testove između fiksne i RL strategije na istim scenarijima.

Robustnost i generalizacija

- **Proširenje scenarija:** uključiti veći spektar seed-ova, različite udjele tipova vozila, incidente, i vremenske uvjete kako bi se smanjio simulation-to-reality gap.
- **Domain randomization i augmentacija:** tijekom treninga nasumično varirati parametre (npr. ponašanje vozača, brzine, udio autobusa) kako bi se polisa naučila biti robusnija.
- **Transfer learning i fine-tuning:** razviti pipeline za finu adaptaciju modela na nove lokacije koristeći manje količine stvarnih ili field-like podataka.

Arhitektura i koordinacija

- **Multi-agent pristup:** istražiti koordinirane ili decentralizirane multi-agentne metode za upravljanje mrežom raskrsnica kako bi se optimiziralo globalno ponašanje saobraćaja.
- **Hierarchical i hybridne metode:** razmotriti hijerarhijske arhitekture (visoki nivo: raspodjela faza; niski nivo: detaljna kontrola trajanja) ili kombinaciju model-based i model-free pristupa.

Analiza i automatizacija eksperimenta

- **Automatizacija hyperparametarskog pretraživanja:** koristiti Bayesian optimization ili grid/random search za traženje optimalnih hiperparametara (alpha, gamma, epsilon decay, arhitektura mreže).
- **Detaljne analize:** pripremiti tabelarni set ključnih statistika (mean, median, std, percentili) po scenariju i po epizodi kako bi se bolje razumjeli uvjeti u kojima agent uspijeva ili ne.

Ove smjernice su namijenjene da prioritetno smanje varijansu i povećaju robusnost naučene polise, te da osiguraju da eventualna primjena u stvarnom svijetu bude sigurna i upotrebljiva.

Prilozi

Ovo poglavlje sadrži dodatne materijale koji podržavaju glavni tekst rada, te na koje se sam rad referira, a zarad preglednosti i svrsishodnog predstavljanja podataka, predstavljeni su kao prilog radu.

Programski kodovi

```
# Identifikacija semafora
TL_ID = "cluster_1955770138_39630061_6970320248_6970320249_#1more"

# Vremenska ograni enja faza
MIN_PHASE_DURATION = 5
MAX_PHASE_DURATION = 110

# Konfiguracija simulacije
CONFIG_FILE = "../simulation-config/osm.sumocfg"
SIMULATION_FOLDER = "../simulation-config/"
NET_FILE = "osm.net.xml"
ROU_FILE = "routes.rou.xml"
SUMO_BINARY = "sumo"
SUMO_BINARY_EVAL = "sumo"

# Parametri treniranja
MAX_STEPS = 22222
EPISODES_DONE = 1500
NUM_EPISODES = 2500
NUM_EVAL_EPISODES = 50
NUM_ROUTE_VARIATIONS = 7

# Hiperparametri Q-u enja
ALPHA = 0.187
GAMMA = 0.95
EPSILON = 1.0
ALPHA_DECAY = 0.9996
EPSILON_DECAY = 0.997

# Putanje za  uvanje  modela
Q_TABLE_PATH = "./q-tables-and-logs/qtable_final.pkl"
EVAL_Q_TABLE_PATH = "q-tables-and-logs/tables/qtable_ep"

# Parametri generisanja ruta
SIM_START_OF_GENERATING = 0
SIM_GENERATING_RANGE_MIN = 800
SIM_GENERATING_RANGE_MAX = 1100
ROUTES_PER_SEC_RANGE_MIN = 0.7
ROUTES_PER_SEC_RANGE_MAX = 1.1
ROUTES_PER_SEC_RANGE_RANDOMIZE = False

last_alpha = 0.060156
last_gamma = 0.950000
last_epsilon = 0.001538
episodes_done = 2199
```

Program 1: Konfiguracijski parametri simulacije

```
class QLearningAgent:
    def __init__(self, actions, alpha=0.1, gamma=0.95, epsilon=0.5):
        self.q_table = {}
        self.alpha = alpha
        self.gamma = gamma
        self.epsilon = epsilon
        self.actions = actions

    def get_state_key(self, state):
        phase, duration, *queues = state

        MAX_QUEUE = 60 # prilagoditi po ocekivanom maksimalnom broju vozila u redu
        QUEUE_STEP = 5

        queue_bins = [min(q, MAX_QUEUE) // QUEUE_STEP for q in queues]
        duration_bin = min(int(duration / 10), 10)

        return (phase, duration_bin) + tuple(queue_bins)

    def get_Q(self, state, action):
        key = (self.get_state_key(state), action)
        return self.q_table.get(key, 0.0)

    def choose_action(self, state):
        if random.random() < self.epsilon:
            return random.choice(self.actions)

        q_values = [self.get_Q(state, a) for a in self.actions]
        max_q = max(q_values)

        max_indices = [i for i, q in enumerate(q_values) if q == max_q]
        return self.actions[random.choice(max_indices)]

    def learn(self, state, action, reward, next_state):
        current_key = (self.get_state_key(state), action)
        current_q = self.get_Q(state, action)

        next_max_q = max([self.get_Q(next_state, a) for a in self.actions])

        new_q = current_q + self.alpha * (reward + self.gamma * next_max_q - current_q)
        self.q_table[current_key] = new_q
```

Program 2: Implementacija Q-learning agenta

```
def run_episode(episode, sim_folder=SIMULATION_FOLDER):
    if not os.path.exists(sim_folder):
        print(f"Direktorijum ’{sim_folder}’ ne postoji!")
        return (0, 0, 0, 0, 0)

    os.chdir(sim_folder)
    seed = episode % NUM_ROUTE_VARIATIONS
    sim_generating_end = generate_random_routes(seed)
    os.chdir("../src")

    traci.start([SUMO_BINARY, "-c", CONFIG_FILE, "--no-warnings", "--no-step-log"])
    step = 0
    last_action_time = 0
    total_reward = 0
    departed_vehicles = 0
    arrived_vehicles = 0
    departures_ended = False
    cumulative_waiting = 0
    measurement_count = 0
    last_phase_change_time = 0

    # Inicijalizacija stanja
    lanes = traci.trafficlight.getControlledLanes(TL_ID)
    state = get_state(TL_ID)

    while step < MAX_STEPS:
        traci.simulationStep()
        step += 1
```

```
        current_departed = traci.simulation.getDepartedNumber()
        current_arrived = traci.simulation.getArrivedNumber()

        if step >= sim_generating_end:
            departures_ended = True

        # Provjera zavr etka simulacije
        if departures_ended:
            if arrived_vehicles+current_arrived >= departed_vehicles+current_departed:
                break

        # Prikupljanje podataka o  ekanju
        waiting_times = [traci.vehicle.getWaitingTime(veh_id) for veh_id in traci.vehicle.
            getIDList()]
        if waiting_times:
            cumulative_waiting += sum(waiting_times)
            measurement_count += len(waiting_times)

        # Izbor akcije
        current_phase = traci.trafficlight.getPhase(TL_ID)
        if current_phase == -1:
            continue

        current_state = get_state(TL_ID)

        if step - last_action_time >= MIN_PHASE_DURATION:
            if step - last_action_time >= MAX_PHASE_DURATION:
                action = 1
            else:
                action = agent.choose_action(current_state)

            if action == 1:
                new_phase = (current_phase + 1) % get_phase_count()
                traci.trafficlight.setPhase(TL_ID, new_phase)
                last_action_time = step
        else:
            action = 0

        reward = calculate_reward(current_state)
        total_reward += reward

        # U enje agenta
        next_state = get_state(TL_ID)
        agent.learn(current_state, action, reward, next_state)

        # update pokrenutih i pristiglih vozila
        departed_vehicles += current_departed
        arrived_vehicles += current_arrived

    traci.close()

    avg_waiting = cumulative_waiting / measurement_count if measurement_count > 0 else 0

    return (total_reward, step, sim_generating_end, arrived_vehicles, avg_waiting)

# Glavna petlja treniranja
for ep in range(EPISODES_DONE + 1, NUM_EPISODES + 1):
    agent.epsilon *= EPSILON_DECAY
    agent.alpha *= ALPHA_DECAY

    print(f"Po etak␣epizode␣{ep}")
    reward, steps, gen_end, arrived, avg_wait = run_episode(ep)

    #  uvanje  Q-tabele
    if ep % 40 == 0 or ep == NUM_EPISODES:
        table_path = f"q-tables-and-logs/tables/qtable_ep{ep}.pkl"
        with open(table_path, "wb") as f:
            pickle.dump(agent.q_table, f)
        print(f"Sa uvana␣Q-tabela:␣{table_path}")

        # Pokreni evaluaciju
        print(f"Pokre em␣evaluaciju␣nakon␣{ep}␣epizoda...")
```

```
        os.system(f"{sys.executable} evaluate_agent.py")

    # Logovanje rezultata
    log_entry = f"{ep},{reward},{gen_end},{steps},{arrived},{avg_wait}\n"
    with open("q-tables-and-logs/log.csv", "a") as log_file:
        if ep == 1:
            log_file.write("Episode,Total Reward,Gen End,Sim End,Arrived Vehicles,Avg Waiting\
                n")
        log_file.write(log_entry)

    print(f"Epizoda {ep} zavr ena: Nagrada={reward:.2f}, Vozila={arrived},  ekanje ={avg_wait
        :.2f}s")

    # A uriranje konfiguracije
    update_config(
        last_alpha=round(agent.alpha, 6),
        last_gamma=round(agent.gamma, 6),
        last_epsilon=round(agent.epsilon, 6),
        episodes_done=ep
    )
```

Program 3: Implementacija treniranja agenta

Literatura

[1] Abdulhai, B., Pringle, R., Karakoulas, G. J., "Reinforcement learning for true adaptive traffic signal control", Journal of Transportation Engineering, Vol. 129, No. 3, 2003, str. 278–285.

[2] Yau, K.-L. A., Qadir, J., Khoo, H. L., Ling, M. H., Komisarczuk, P., "A survey on reinforcement learning models and algorithms for traffic signal control", ACM Computing Surveys (CSUR), Vol. 50, No. 3, 2017, str. 1–38.

[3] Balaji, P., German, X., Srinivasan, D., "Urban traffic signal control using reinforcement learning agents", IET Intelligent Transport Systems, Vol. 4, No. 3, 2010, str. 177–188.

[4] Koonce, P. *et al.*, "Traffic signal timing manual", United States. Federal Highway Administration, Tech. Rep., 2008.

[5] Mannion, P., Duggan, J., Howley, E., "An experimental review of reinforcement learning algorithms for adaptive traffic signal control", Autonomic road transport support systems, 2016, str. 47–66.

[6] Goralija, H., Adaptive Traffic Signal Control Repository, 2025, dostupno na: https://github.com/goralija/AdaptiveTrafficSignalControl